МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ
ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____ Аліна САВЧЕНКО
«___»_____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТР
ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ
«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ»

Тема: «Нейронні мережі в data engineering з використанням наборів тренувальних даних»

Виконавець: Максим МАСНИЙ

Керівник: к.т.н., доцент Олена ТОЛСТІКОВА

Нормоконтролер: к.т.н., доцент Олена ТОЛСТІКОВА

КИЇВ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *кібербезпеки, комп'ютерної та програмної інженерії*
Кафедра *Комп'ютерних інформаційних технологій*
Спеціальність *122 «Комп'ютерні науки»*
Освітньо-професійна програма *«Інформаційні технології проектування»*

ЗАТВЕРДЖУЮ:
завідувач кафедри КІТ
_____Аліна САВЧЕНКО
(підпис)

«____»_____2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи
Масного Максима Вікторовича
(ПІБ випускника)

1. Тема роботи: «Нейронні мережі в data engineering з використанням наборів тренувальних даних» затверджена наказом ректора № 1774/ст від 28.09.2022р.
2. Термін виконання роботи: з 26 вересня 2022 року по 21 листопада 2022 року.
3. Вихідні дані до роботи: навчена згорткова нейронна мережа з використанням створених наборів даних.
4. Зміст пояснювальної записки: 1. Огляд та аналіз предметної області. 2. Розгляд та аналіз нейронної мережі YOLO. 3. Аналіз використовуваних технологій. 4. Навчання та тестування нейронної мережі.
5. Перелік обов'язкового ілюстративного матеріалу: 1. Актуальність. 2. Мета кваліфікаційнох роботи. 3. Практична цінність. 5. Об'єкт дослідження. 6. Штучний інтелект. 7. Вибір архітектури нейронної мережі. 8. Архітектура YOLOv5. 9. Принцип роботи нейронної мережі із зображеннями. 10. Послідовність розробки системи комп'ютерного зору. 11. Результат роботи навченої нейронної мережі. 12. Висновки.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Аналіз предметної області та огляд аналогів. Написання 1 розділу, представлення керівнику	26.09.2022- 05.10.2022	
2.	Огляд обраної архітектури на підставі аналізу доступних версій. Написання 2 розділу, представлення керівнику	06.10.2022- 18.10.2022	
3.	Розгляд доступних методів реалізації та аналіз використовуваних технологій. Написання 3 розділу, представлення керівнику	19.10.2022- 30.10.2022	
4.	Тренування та валідація нейронної мережі. Написання 4 розділу, представлення керівнику.	31.10.2022- 14.11.2022	
5.	Загальне редагування та друк пояснювальної записки	15.11.2022- 20.11.2022	
6.	Проходження нормоконтролю, перепліт пояснювальної записки.	16.11.2022- 20.10.2022	
7.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації	20.11.2022- 22.11.2022	

7. Дата видачі завдання 26.09.2022р.

Керівник кваліфікаційної роботи _____ Олена ТОЛСТІКОВА
(підпис керівника)

Завдання прийняв до виконання _____ Максим МАСНИЙ
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему: «Нейронні мережі в data engineering з використанням наборів тренувальних даних» містить: 100 сторінок, 60 рисунків, 3 таблиці, 40 інформаційних джерел, 5 додатків.

Об'єкт дослідження – система комп'ютерного бачення для знаходження об'єктів, розроблена на мові програмування Python.

Предмет дослідження – використання наборів даних у навчані згорткових нейронних мереж.

Мета кваліфікаційної роботи – отримати готову систему штучного інтелекту для знаходження об'єктів на зображеннях за допомогою використання наборів тренувальних даних.

Методи дослідження – мова програмування Python, інтегроване середовище розробки Google Colab**,** архітектура нейронної мережі YOLOv5, OpenCV, CVAT, FiftyOne, NumPy.

Результати кваліфікаційної роботи рекомендується використовувати при створенні автоматизованої системи підготовки тренувальних даних.

Для розроблення системи комп'ютерного зору було використано мову програмування Python, методи створення навчальних та валідаційних даних, шаблони тренування згорткових нейронних мереж, використання методів реалізації комп'ютерного зору, валідація отриманої нейронної мережі.

ШТУЧНИЙ ІНТЕЛЕКТ, ШАБЛОН ПРОЕКТУВАННЯ, КОМП'ЮТЕРНИЙ ЗІР, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, НАВДАЛЬНІ НАБОРИ ДАНИХ, ДАТАСЕТ, ТРЕНУВАННЯ ТА ВАЛІДАЦІЯ МОДЕЛЕЙ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

AI *(Artificial Intelligence)*	—	Штучний інтелект
DL *(Deep Learning)*	—	Глибоке навчання
CNN *(Convolution Neural Network)*	—	Згорткова нейронна мережа
ML (Machine Learning)	—	Машинне навчання
RNN *(Recurrent Neural Network)*	—	Рекурентна нейронна мережа
CV *(Computer Vision)*	—	Комп’ютерний зір
SIFT(Scale-invariant Feature transform)		Масштабно-інваріативне перетворення ознак
MVP *(Minimal valuable product)*	—	Мінімально життєздатний продукт
IoU *(Intersection over Union)*	—	Перетин через об’єднання
YOLO *(You Only Look Once)*	—	«Ти проглядаєш тільки один раз», назва нейронної мережі

ВСТУП

У середині 1950-х років було розроблено перше програмне забезпечення, яке по функціоналу було наближче до людських навичок і вирішення проблем. Хоча концепція штучного інтелекту еволюціонувала протягом останніх двадцяти років, проблема полягала не в програмуванні, а у відсутності пам'яті комп'ютера, що сповільнювало його роботу. Для розробки штучного інтелекту потрібні були вищі швидкості обробки інформації, особливо у випадку комп'ютерного зору.

Програми, які вимагають більше обчислювальної потужності, ніж більшість сучасних пристроїв, є чудовими кандидатами на системи ШІ. Наприклад, у сфері охорони здоров'я машинне навчання може групувати схожі медичні випадки, щоб ідентифікувати закономірності, використовувати найновіші дані для прогнозування та виконувати інші завдання з керування даними. Комп'ютерний зір – це галузь штучного інтелекту, яка використовує дані для виявлення та ідентифікації об'єктів, які були знайдені через камеру девайсу [1].

Актуальність теми кваліфікаційної роботи «Нейронні мережі в data engineering з використанням наборів тренувальних даних» ґрунтується на тому, що в час розвитку штучного інтелекту та комп'ютерного зору збільшується необхідність в генеруванні великої кількості даних та навчальних матеріалів. Саме це провокує проводити дослідження, розробляти нові рішення, щоб створити системи, які надалі будуть в автоматизованому режимі генерувати нові датасети. На даний час данні є одним з головніших факторів, які можуть вплинути на покращення роботи нейронних моделей та на збільшити їх точність.

Кожна ітерація розробки в сфері штучного інтелекту веде до нових відкритів або обмежень, які частіше відбуваються через не змогу подолати певні програмні ліміти. Тому в ШІ данні допомагають вирішити питання, що призводять до нового кроку вперед. Особливо, якщо розглядати згорткові нейронні мережі та комп'ютерний зір, які стають все більш популярні у

рішення звичайних буденних задач або при достатньо складних системах для задач великої компанії.

Метою кваліфікаційної роботи є розробка системи комп'ютерного зору для знаходження об'єктів за допомогою створених тренувальних даних.

Відповідно до поставленої мети роботи визначено основні **завдання дослідження**:

– провести аналіз наукової та методичної літератури;

– проаналізувати системи комп'ютерного зору та рішення, які вже існують на ринку;

– провести огляд методологій підготовки та знаходження даних, створення наборів даних та використання їх при навчанні нейронних мереж;

– розробити систему комп'ютерного бачення для розпізнавання об'єктів на зображеннях;

– описати принцип створення ланцюга підготовки даних та використання його у навчальних процесах.

Для досягнення поставленої мети й виконання завдань використано метод системно-структурного аналізу наукової літератури, який дав змогу показати особливості створення та використання даних у тренуванні штучного інтелекту. Для формулювання і систематизації висновків використано методи аналізу, формалізації, абстрагування та узагальнення.

Наукова новизна кваліфікаційної роботи полягає в виділенні позитивних та негативних рішень для створення та перевірки датасетів, актуалізацію розробленого ланцюга автоматичного створення даних у подальших навчання згорткових нейронних мереж.

Розроблена система має перспективу користуватись великим попитом у зв'язку з широкою доступністю, зручністю та зрозумілістю для кожного інженера даних. Цей інструмент допоможе команді по розробці рішення ШІ збільшити швидкість обробки великої кількості інформації та прискорити отримання необхідного результату при навчанні. У майбутньому планується покращувати модель CV за рахунок збільшення навчальних даних.

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Машинне навчання

Машинне навчання — це розробка алгоритмів і пов’язаних систем, які можуть навчатися стратегіям поведінки в певних середовищах за допомогою інструкцій і навчальних наборів даних.

Будучи підмножиною штучного інтелекту (ШІ), машинне навчання значною мірою обходить деякі ширші філософії ШІ та допомагає визначити підходи до системного навчання, які можуть створювати ефективні машини для будь-якого середовища. Ця дисципліна зосереджена на статистичних моделях, алгоритмах і методах учнівства, які розробляють машини в секторах австралійських водолазів, які займаються виробництвом, продажами або роздрібною торгівлею, логістикою ланцюга схвалення, перетворенням продуктів харчування та будівництвом[2].

Різноманітні підходи до машинного навчання підкреслюють алгоритми розпізнавання шаблонів у даних для інформування про стратегічні дії в подібних середовищах. Ці підходи включають наступне:

1. Навчання з учителем (Supervised Learning) – це категорія навчання моделі, у якій ми подаємо позначені навчальні набори даних, іншими словами датасети, під наглядом спеціаліста, із каталогом вхідних даних і відповідних бажаних результатів. Вхідні та вихідні значення завідомо вже відомі, а сам механізм штучного інтелекту вивчає функцію відображення. Математично, маючи Y, як результат, та X як вхідні дані, алгоритми штучного інтелекту намагаються знайти найоб’єктивнішу функцію відтворення f таку, що Y = f(X).

Якщо розглянути це питання, то навчання відбувається так, ніби спеціаліст з даних контролює процес засвоєння моделі. Алгоритми намагаються відобразити функцію таким чином, щоб predictions були близькі до фактичних правдивих значень. Допустимо, що система навчилася відтворювати функції f, яка передбачає значення Y' для кожного X, тоді різниця між очікуваним (Y') і фактичним (Y) падає нижче певного порогу, помилки залишаються, хоча й незначні. Це означає, що показники моделі не покращуються, а навчання припиняється. З учителем машина вчиться набагато краще й швидше, тому для вирішення практичних завдань такі алгоритми використовують частіше. До алгоритмів навчання з учителем належать такі типи задач, як регресія і класифікація [3].

Прикладом класифікації може слугувати модель, яка аналізує геометричні фігури на прикладі нижче (рис. 1.1). Зображення форми є розміченими вхідними даними для моделі, а розмітка цих зображень як їх опис, цей процес називається анотацією, є результуючими даними. Покладаючись на ці вхідні і вихідні дані система штучного інтелекту навчається прогнозувати вид отримуваного зображення у відповідності до реального опису фігур.

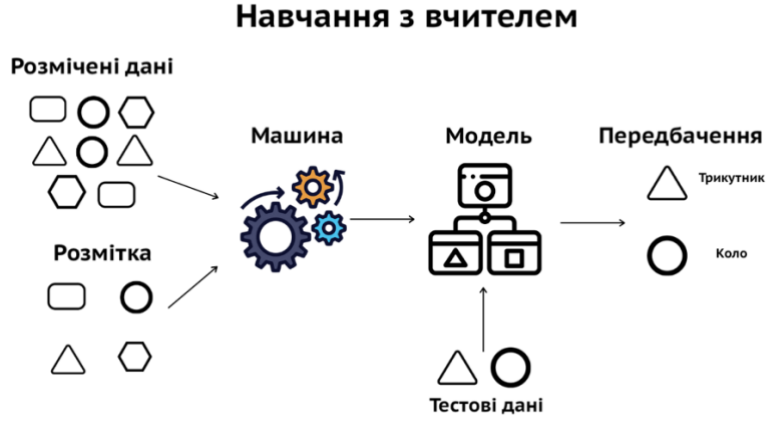


Рис. 1.1. Реалізація методу навчання з вчителем

Регресія, на прикладі зображення нижче, на осі X відображено досвід - у роках. Маючи кількість років досвіду для кожного з показників на осі Y

нараховується відповідна заробітна плата в українських гривнях. Зелені крапки – це координати (X, Y) у формі вхідних і вихідних даних. Задача регресії намагається віднайти функцію безперервного відтворення на основі вхідних даних так, щоб отримати точки на графіку відповідно до вихідних змінних. На зображенні нижче (рис. 1.2), якщо функція відображення першого порядку, що є лінійною, то модель вивчатиме відображення чорної лінії.

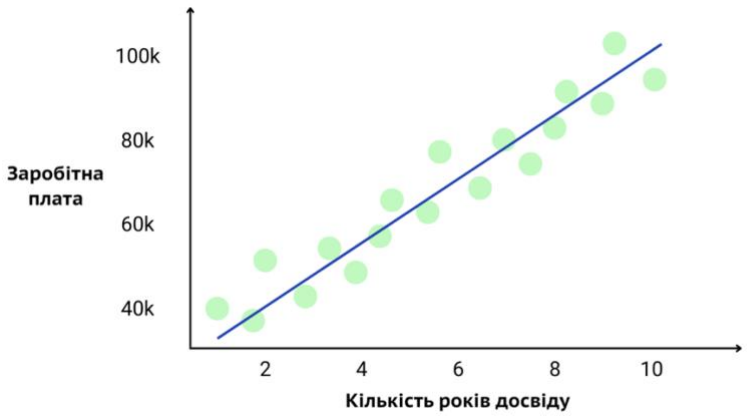


Рис. 1.2. Реалізація регресії

Найпоширеніші приклади використання мотоду навчання з учителем:

1) Детект об’єктів і класифікація зображень, де визначається, чи є шуканий системою об’єкт на зображенні, якщо є, то потрібно знайти його розташування;

2) Системи пошукових рекомендацій: маючи параметри пошуку від клієнта, система прогнозує релевантні його запиту та бажанням результат відштовхуючись від даних, які узагальнюються по всій виборці.

3) Прогноз часових рядів: маючи певну інформацію, наприклад про температуру, погодні умови, періодичні дії чи можливо коливання на фондовому ринку або будь-яке циклічне відтворення подій, спрогнозувати зміну та поведінку середовища.

У навчанні під керівництвом зазвичай використовують такі алгоритми для побудови системи машинного навчання:

- лінійна та логістична регресія;

- SVM (машини опорних векторів);
- випадковий ліс.

2. Навчання без вчителя (Unsupervised Learning), як легко зрозуміти з назви, це підходи до неконтрольованого навчання, які використовують неструктуровані набори даних без пов’язаних ідеальних результатів. Це категорія машинного навчання, де ми маємо лише вхідні дані для подачі моделі, але не маємо відповідних вихідних даних. Тут ми маємо значення вхідних даних, але результат та функція відтворення невідомі. Тоді система штучного інтелекту знаходить функцію, яка має зрозуміти подібності між зразками різних вхідних даних, шаблонів і групує їх відповідно до індексу подібності, який є результуючим значенням, тобто розробити стратегії поведінки. Іншими словами, ми можемо сказати, що алгоритми створюють псевдовихід для вивчення функції відповідності. Зазвичай таким підходом рідко користуються, частіше як метод аналізу та підготовки даних, а не як основний алгоритм, що знаходить конкретне рішення на основі цих даних.

На сьогодняшній день датасети з добре анотованими даними – це велика знахідка, тому що проце розмітки трудоємкий. Для цього необхідно застосовувати або профільні сервіси для анотації, або ж підготовкою даних займається відповідна команда людей, або спеціальні алгоритми, зазвичай це також алгоритми машинного навчання. Проте найдієвіший варінт – це поєднання автоматичної розмітки за допомогою інших моделей штучного інтелекту у поєднанні з людським ресурсом. Це допоможе пришвидшити команду анотації та покращити точність мережі.

Навчання без вчителя можна класифікувати на:

1) Кластеризація, неконтрольована класифікація. Використовуючи наведений нижче приклад (рис. 1.3), формується нерозмічений набір даних на вході, що складаються із зображень фігур. Модель штучного інтелекту отримує елементи певних форм та намагається віднайти схожість між вхідною інформацією та сформувати правила групування за певним розміром, характеризуючими рисами або на основі піксельних значень кольору, по яким

кластеризуються вихідні дані, що містять подібні вхідні зразки. В даному методі розповсюджені такі алгоритми, а саме ієрархічне групування та метод К-середніх.

Рис. 1.3. Реалізація методу навчання без вчителя

2) Зменшення розмірності: коли атрибути зразків даних мають більше трьох вимірів, неможливо візуалізувати зв’язок між атрибутами, оскільки ми не можемо представити змінні більш ніж у трьох вимірах. Але ми ніколи не можемо бути впевнені в продуктивності моделі машинного навчання без аналізу вхідних даних. Для досягнення цієї мети існують методи зменшення розміру, які використовуються для підсумовування загальної кількості вимірів і аналізу даних. Припустімо, необхідно проаналізувати 10 функцій разом, але немає можливості відобразити 10D-графіки. Тому варто зменшити розмірність до 3D або меншої, щоб далі відслідковувати, аналізувати та опрацьовувати їхні зв’язки.

3) Асоціювання - це приклад використання анотованих та не анотованих даних, що в комбінаціїї допомагає машині створити правила для визначення об’єктів. Це працює схоже на систему рекомендацій, якщо клієнт купує товар А, то йому може бути корисним товар С, тому що ця комбінація користується попитом (рис. 1.4). В наш час добре відомі випадки застосування навчання без вчителя у таких ситуаціях:

- сегментація ринку - гарячий чи холодний ринок залежить від грошей, що на ньому обертаються;

- виявлення шахрайства - роздіження транзакції на нешахрайські групи;
- сегментація зображення, наприклад - сірі об’єкти на зображенні, зробленому автомобілем, представляють дороги.

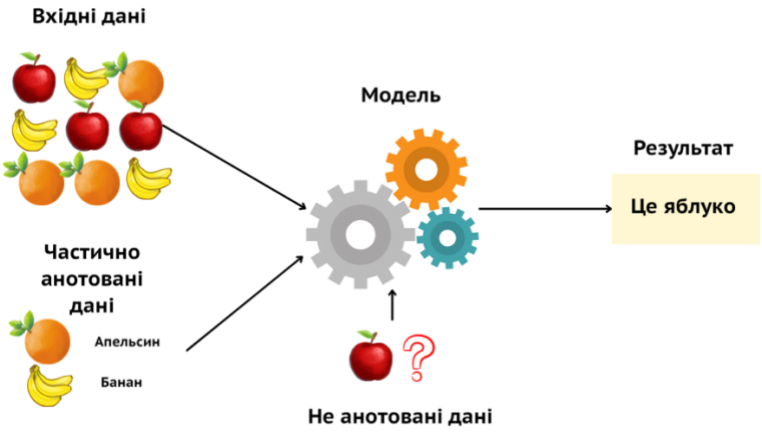


Рис. 1.4. Алгоритм асоціювання

Нижче наведено деякі поширені алгоритми, які використовуються в неконтрольованому навчанні: k-means, апріорний алгоритм у навчанні правил асоціації, аналіз головних компонентів.

3. Навчання з підкріпленням (Reinforcement Learning) – це навчання, яке зазвичай використовується для навчання незалежних машинних агентів у певній системі, використовує моделі кумулятивних винагород, щоб навчити агентів діяти в різних системах. Алгоритми стежать за максимізацією винагороди та зменшенням ризику та, зрештою, навчанням. Це застосування МН використовується в промислових галузях, проте зараз набирає популярності та стає широко вживаною в багатокористувацьких онлайн-іграх.

Середовищем може бути що завгодно: як реальний світ, так і симуляція, і навіть комп'ютерна гра. Наприклад, існують роботи, які навчилися грати в Dota2 просто поринувши в середовище, або автопілот Tesla, який у симуляції вчиться не збивати пішоходів. Також дуже відомий приклад, де нейронна мережа AlphaGo зіграла сама з собою тисячі разів у відому гру Go, яка в свою чергу дуже складна та має мільйони комбінацій ходів, а в результаті виробила

стратегію. Саме це допомогло штучному інтелекту виграти у тодішнього чемпіона з Go – Лі Седоля.

Знання про довкілля таким роботам корисні, але знати весь світ їм не обов'язково, тому що вони можуть генерувати та симулювати ситуації всередині себе. Завдання таких машин – не прораховувати всі можливі ходи, а мінімізувати loss або максимізувати score. Цей вид навчання є максимально схожим на реальний розвиток людей – машину карають за помилки і заохочують за правильні вчинки (рис. 1.5).

Рис. 1.5. Реалізація методу навчання з підкріпленням

Успіхи у розроблених систем в комп'ютерних іграх, спорті або технології автопілоту машини мають ефектний вигляд, проте насправді алгоритмів reinforcement learning мало. Цей напрямок розвиватися та тільки набирає обертів. аме тому за ними, безсумнівно, майбутнє.

4. Традиційно системи машинного навчання та ШІ використовували лінійні або ітераційні підходи до машинного навчання. У 1980-х роках і далі дослідники розробили мозок «нейронної мережі», використовуючи структури вузлів і кластерів і зважені стратегії прийняття рішень. Таким чином, системи машинного навчання можуть розбивати складні проблеми на простіші, а результати більш простих проблем можуть об’єднуватися в більш комплексне рішення для більших, що називається глибоким навчанням та нейронними мережами.

Глибоке навчання зробило ще один крок далі, створивши нейронні мережі на основі рівнів, де рівні мереж, заснованих на рішеннях, могли по суті створити механізм вирішення проблем. Наприклад, мозок із глибоким навчанням може мати рівні, де прості підходи до розпізнавання образів можуть об'єднуватися для виконання складних завдань, таких як розпізнавання обличчя на зображеннях.

В усіх підходах до машинного навчання наголос завжди робиться на тому, як навчати системам машинного навчання, симулювати навчальні середовища для машинного навчання та використовувати машинне навчання для забезпечення комплексного штучного інтелекту та автономних систем [4].

1.2. Комп'ютерний зір (Computer Vision)

Комп'ютерне бачення використовує штучний інтелект, щоб дозволити комп'ютерам отримувати важливу інформацію з візуальних вхідних даних, таких як фотографії та відео. Дані, отримані за допомогою комп'ютерного зору, потім використовуються для виконання автоматизованих дій. Подібно до того, як ШІ дає комп'ютерам здатність «мислити», комп'ютерний зір дозволяє їм «бачити» (рис 1.5).

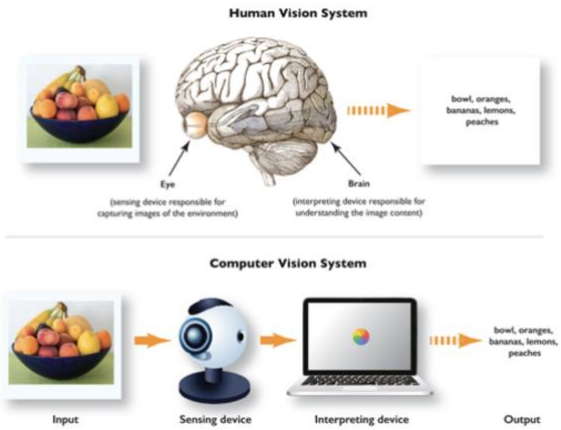


Рис. 1.5. Принцип роботи людського та комп'ютерного зору [5]

Люди зазвичай проводять своє життя, спостерігаючи за навколишнім середовищем за допомогою зорових нервів, сітківки та зорової кори. Отримують контекст, щоб розрізняти об'єкти, вимірювати їхню відстань від

нас та інших об’єктів, обчислювати швидкість їх руху та виявляти помилки. Подібним чином комп’ютерний зір дозволяє машинам на основі штучного інтелекту навчитися виконувати ці самі процеси. Ці машини використовують для цього комбінацію камер, алгоритмів і даних.

Однак, на відміну від людини, комп’ютер не втомлюється. Ви можете навчити машини з комп’ютерним зором аналізувати тисячі виробничих активів або продуктів за лічені хвилини. Це дозволяє виробничим підприємствам автоматизувати виявлення дефектів, непомітних для людського ока.

Щоб комп’ютерний зір був справді ефективним, потрібна велика база даних. Це пояснюється тим, що ці рішення аналізують інформацію неодноразово, доки не отримають усі можливі відомості, необхідні для виконання поставленого завдання. Наприклад, комп’ютер, навчений розпізнавати здорові посіви, мав би «бачити» тисячі візуальних еталонних даних про посіви, сільськогосподарські угіддя, тварин та інші пов’язані об’єкти. Лише тоді він зможе ефективно розпізнавати різні типи здорових культур, диференціювати їх від нездорових, оцінювати якість сільськогосподарських угідь, виявляти шкідників та інших тварин серед культур тощо.

Дві ключові технології керують комп’ютерним зором: згорточна нейронна мережа та глибоке навчання, тип машинного навчання. Машинне навчання використовує моделі на основі алгоритмів, щоб дозволити комп’ютерам вивчати контекст за допомогою візуального аналізу даних. Як тільки достатньо даних буде надано моделі, вона зможе «бачити загальну картину» та розрізняти візуальні вхідні дані. Замість того, щоб бути запрограмованим на розпізнавання та розрізнення зображень, машина використовує алгоритми ШІ для автономного навчання [6].

Згорткові нейронні мережі допомагають моделям МН бачити, розділяючи зображення на пікселі. Кожному пікселю надається мітка або тег. Потім ці мітки спільно використовуються для виконання згорток,

математичного процесу, який поєднує дві функції для створення третьої функції. Завдяки цьому процесу згорткові нейронні мережі можуть обробляти візуальні вхідні дані.

Щоб бачити зображення так само, як людина, нейронні мережі виконують згортки та перевіряють точність результату в численних ітераціях. Подібно до того, як люди бачать об'єкт на великій відстані, згорточна нейронна мережа починається з ідентифікації рудиментарних форм і жорстких країв. Коли це зроблено, модель виправляє прогалини у своїх даних і виконує ітерації свого виводу. Це триває доти, доки результат точно не «передбачить», що станеться.

У той час як згорточна нейронна мережа розуміє окремі зображення, рекурентна нейронна мережа обробляє відеовхідні дані, щоб дозволити комп'ютерам «дізнаватися», як серія зображень пов'язана одна з одною.

Оскільки штучний інтелект і машинне навчання відновилися в останні роки завдяки високопродуктивним обчисленням, великим даним і хмарним системам, також зріс попит на апаратне забезпечення, яке може підтримувати візуальний збір даних для програм машинного навчання (рис. 1.6).

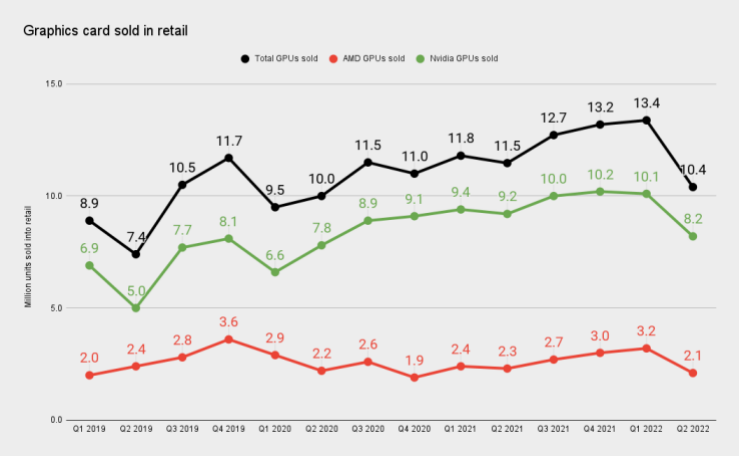


Рис. 1.6. Ріст попиту на обчислювальне апаратне забезпечення, тобто GPU [7]

Як і більшість систем машинного навчання, комп'ютерне зір потребує значної кількості даних для навчання алгоритмів інтерпретації цих даних.

Deep Learning як згадувалося раніше, може сприяти вирішуванню складних проблем. Що ще важливіше, глибоке навчання з використанням нейронних мереж може по суті навчити «мізки» машин сприймати візуальні дані та зберігати знання про шаблони, стратегії та зміни змінних навколишнього середовища з часом.

Згорткові нейронні мережі приймають візуальну інформацію, як зображення, і розбивають її на пікселі, використовуючи «згортки», операція створення математичної функції з двох інших функцій, щоб робити прогнози щодо цих даних.

По суті, комп'ютерний зір використовує CNN і глибоке навчання для виконання високошвидкісного неконтрольованого навчання великої кількості візуальної інформації, щоб навчити системи машинного навчання інтерпретувати дані у спосіб, який дещо нагадує роботу людського ока.

1.3. Як машинне навчання охоплює комп'ютерний зір?

Комп'ютерний зір є підмножиною машинного навчання. Після того, як у середині 1980-х до середини 1990-х років інтерес до досліджень штучного інтелекту та машинного навчання знизився, більшість розробок у цій галузі розділилася на такі підполя, як обробка природної мови, розпізнавання зображень і робототехніка.

Технології ніколи не перестають імітувати людський мозок, тому ШІ викликає великий інтерес протягом десятиліть. Крім того, комп'ютерний зір можна визначити як підмножину глибокого навчання. Замість того, щоб обробляти змодельовані дані чи статистичні дані, комп'ютерний зір руйнує та інтерпретує візуальну інформацію.

Машинне навчання та комп'ютерний зір – це дві сфери, які тісно пов'язані одна з одною. МН покращило комп'ютерне бачення щодо розпізнавання та відстеження. Він пропонує ефективні методи отримання, обробки зображень і фокусування об'єктів, які використовуються в комп'ютерному зорі. У свою чергу комп'ютерний зір розширив сферу

застосування машинного навчання. Він включає цифрове зображення або відео, сенсорний пристрій, пристрій інтерпретації та етап інтерпретації. Машинне навчання використовується в комп’ютерному зорі в пристрої інтерпретації та на етапі інтерпретації.

Важливо відзначити, що комп’ютерний зір не є необхідним у багатьох програмах машинного навчання. Система машинного навчання, яка керує виробничою лінією або моделює цифрові близнюки для суднових танкерів, не має великої користі від можливостей комп’ютерного зору. Інформація, необхідна цим системам для вивчення та роботи, доступна у вигляді числових представлень.

З іншого боку, системи комп’ютерного зору потребують візуальної інформації для навчання та функціонування. Системи комп’ютерного бачення поєднуватимуть підходи до машинного навчання, які раніше обговорювалися, з такими апаратними засобами, як камери, оптичні датчики тощо. Цей підхід надає певні обмеження, включаючи проблеми з апаратним забезпеченням і способами перетворення зображень у корисні структури даних для машинного навчання.

Відносно, машинне навчання є ширшою сферою, і це очевидно в алгоритмах, які можна застосовувати в інших сферах. Прикладом є аналіз цифрового запису, який виконується з використанням принципів машинного навчання. Комп’ютерне зір, з іншого боку, в основному має справу з цифровими зображеннями та відео. Крім того, він має стосунки в галузі інформаційної інженерії, фізики, нейробіології та обробки сигналів.

Перешкодою, з якою стикаються розробники та підприємці, є величезний розрив між комп’ютерним та біологічним зором. Сфери, найбільш тісно пов'язані з комп'ютерним зором, - це обробка та аналіз зображень. Однак він заслуговує ще однієї цікавої статті, щоб процитувати його взаємозв’язок і відмінності. Крім того, відсутність знань про основну мету машинного навчання в конкретному проекті викликає величезний розлад серед підприємців.

Незважаючи на ці проблеми, комп’ютерне бачення було реалізовано в кількох контекстах:

1) Самокеровані автомобілі, де системи машинного навчання мають збирати візуальні дані для безпечного керування транспортним засобом.

2) Роздрібна торгівля та інвентаризація, де вдосконалені камери в магазинах Amazon Go використовувалися для відстеження того, коли фізичні товари видаляються або замінюються з полиць, щоб оновлювати онлайн-інвентаризаційні запаси, одночасно спрощуючи процес оформлення замовлення.

3) Охорона здоров’я, де зображення крові на хірургічних інструментах можна використовувати для оцінки крововтрати та надання хірургам точної інформації про стан пацієнта.

1.4. Завдання з комп’ютерним зором

Розпізнавання в комп’ютерному зорі передбачає розпізнавання, ідентифікацію та виявлення об’єктів. Деякі спеціалізовані завдання розпізнавання – це оптичне розпізнавання символів, пошук зображень і розпізнавання обличчя.

Розпізнавання об’єктів – передбачає пошук та ідентифікацію об’єктів на цифровому зображенні чи відео. Найчастіше він використовується для виявлення та розпізнавання облич. До розпізнавання об’єктів можна підійти за допомогою машинного або глибокого навчання.

Підхід машинного навчання – розпізнавання об’єктів за допомогою машинного навчання вимагає визначення ознак перед класифікацією. Поширеним підходом, який використовує машинне навчання, є масштабно-інваріантне перетворення ознак (SIFT). SIFT використовує ключові точки об'єктів і зберігає їх у базі даних. Під час класифікації зображення SIFT перевіряє ключові точки зображення, які відповідають тим, що містяться в базі даних.

Підхід глибокого навчання – розпізнавання об’єктів за допомогою глибокого навчання не потребує спеціально визначених функцій. Загальні підходи, які використовують глибоке навчання, базуються на згорткових нейронних мережах. Згорткова нейронна мережа — це тип глибокої нейронної мережі, яка є штучною нейронною мережею з кількома рівнями між входом і виходом. Штучна нейронна мережа — це обчислювальна система, створена за мотивами біологічної нейронної мережі в мозку. Найкращим прикладом цього є ImageNet. Це візуальна база даних, призначена для розпізнавання об’єктів, продуктивність якої майже подібна до людської [8].

Аналіз руху в комп’ютерному зорі передбачає цифрове відео, яке обробляється для створення інформації. Проста обробка може виявити рух об'єкта. Більш складна обробка відстежує об’єкт у часі та може визначити напрямок руху. Його можна застосувати для захоплення руху, спорту та аналізу ходи.

Захоплення руху – передбачає запис руху об’єктів. Маркери наносять біля суглобів, щоб визначити рух. Він має застосування в анімації, спорті, комп’ютерному зорі та аналізі ходи. Як правило, записуються лише рухи акторів, а візуальний вигляд не включається (рис. 1.7).

Рис. 1.7. Приклад захоплення руху людини під час тренування [9]

Аналіз ходи - це дослідження локомоції та діяльності м'язів за допомогою інструментів. Він передбачає кількісну оцінку та інтерпретацію

моделі ходи. Необхідно кілька камер, підключених до комп'ютера. Суб'єкт носить маркери на різних опорних точках тіла. Під час руху об’єкта комп’ютер розраховує траєкторію кожного маркера в трьох вимірах. Його можна застосувати до спортивної біомеханіки.

1.5. Сфери розвитку комп’ютерного бачення

Відеоспостереження – це процес визначення місця розташування рухомого об’єкта в часі. Розпізнавання об’єктів використовується для допомоги у відстеженні відео. Відеоспостереження можна використовувати в спорті. Спорт передбачає багато рухів, і ці технології ідеально підходять для відстеження рухів гравців.

Автономні транспортні засоби – комп’ютерне бачення використовується в автономних транспортних засобах, таких як самокерований автомобіль. Камери розміщені на верхній частині автомобіля, що забезпечує поле огляду на 360 градусів на відстані до 250 метрів. Камери допомагають знаходити смугу руху, оцінювати кривизну дороги, виявляти перешкоди, дорожні знаки та багато іншого. Комп'ютерний зір має реалізувати виявлення та класифікацію об'єктів.

Комп’ютерний зір використовується у спорті для покращення досвіду трансляції, підготовки спортсменів, аналізу та інтерпретації та прийняття рішень. Спортивна біомеханіка - це кількісне дослідження та аналіз спортсменів і видів спорту. Для покращення трансляції можна намалювати віртуальні маркери по полю чи майданчику. Що стосується підготовки спортсмена, то створення моделі скелета акробата та оцінка центру маси дозволяє покращити форму та поставу. Нарешті, для спортивного аналізу та інтерпретації гравці відстежуються в живих іграх, що дозволяє отримувати інформацію в реальному часі.

CV використовується для отримання даних для досягнення аналітики баскетболу. Ці аналітичні дані витягуються за допомогою відстеження відео та розпізнавання об’єктів шляхом відстеження рухів гравців. Методи аналізу

руху також використовуються для допомоги у відстеженні руху. Для аналізу даних використовується глибоке навчання за допомогою згорткових нейронних мереж.

Візьмемо, наприклад, Second Spectrum — офіційного партнера НБА (Національна баскетбольна асоціація) з відстеження — щодо процесу розробки програмного забезпечення. Second Spectrum використовує великі дані, машинне навчання та комп'ютерне зір, щоб забезпечити аналітику та створити машини, які розуміють спорт. Система використовує дані оптичного відстеження та з'ясував, що триочкові та кидки зблизька ефективніші, ніж кидки із середньої дистанції. Крім того, було виявлено, що потенційні підбирання згруповані поблизу кошика [10].

1.6. CNN - основа сучасного комп'ютерного зору

Сучасні алгоритми комп'ютерного зору базуються на згорткових нейронних мережах, які забезпечують різке покращення продуктивності порівняно з традиційними алгоритмами обробки зображень.

CNN зазвичай використовуються для завдань комп'ютерного зору, хоча аналіз тексту та аудіо також можна виконувати. Однією з перших архітектур CNN була AlexNet, яка виграла конкурс візуального розпізнавання ImageNet у 2012 році.

Коли зображення обробляється CNN, кожен базовий колір, який використовується в зображенні, наприклад, червоний, зелений, синій, представляється як матриця значень. Ці значення оцінюються та конденсуються в тривимірні тензори у випадку кольорових зображень, які являють собою набори стеків карт функцій, прив'язаних до частини зображення.

Ці тензори створюються шляхом проходження зображення через серію згорткових і об'єднуючих шарів, які використовуються для вилучення найбільш релевантних даних із сегмента зображення та зведення їх у меншу репрезентативну матрицю. Цей процес повторюється кілька разів (залежно від

кількості згорткових шарів в архітектурі). Остаточні характеристики, витягнуті за допомогою згорткового процесу, надсилаються на повністю пов’язаний рівень, який генерує прогнози.

Продуктивність і ефективність CNN визначається її архітектурою. Це включає структуру шарів, те, як елементи розроблені та які елементи присутні в кожному шарі. Було створено багато CNN, але нижче наведено деякі з найефективніших проектів:

1. AlexNet (2012) - це архітектура, заснована на попередній архітектурі LeNet. Він включає п'ять згорткових шарів і три повністю з'єднані шари. AlexNet використовує структуру подвійного конвеєра, щоб забезпечити використання двох графічних процесорів під час навчання (рис. 1.8).

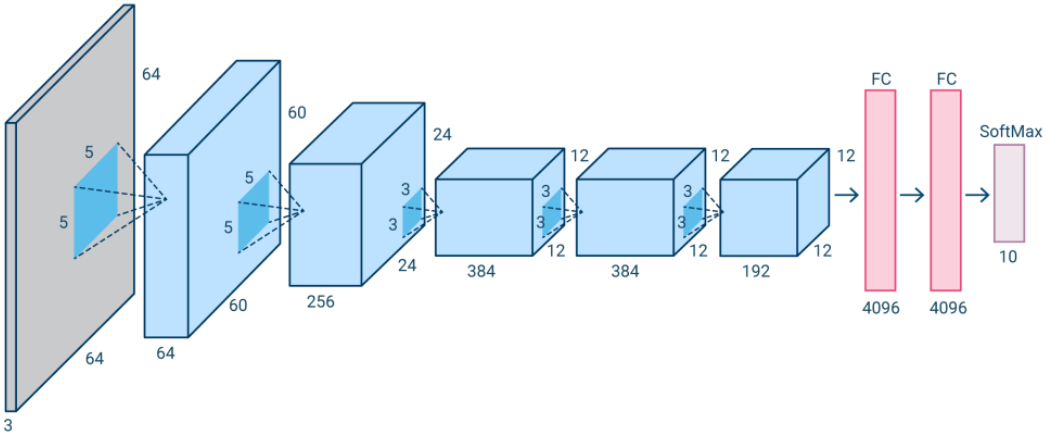


Рис. 1.8. Приклад архітектури нейронної мережі AlexNet [11]

Основною відмінністю між AlexNet і попередніми архітектурами є використання випрямлених лінійних одиниць (ReLU) замість сигмоїдних або активаційних функцій Tanh, які використовувалися в традиційних нейронних мережах. ReLU є простішим і швидшим для обчислень, що дозволяє AlexNet швидше навчати моделі.

2. GoogleNet (2014) - також відомий як Inception V1, базується на архітектурі LeNet. Він складається з 22 шарів, які складаються з невеликих груп звивин, які називаються «початковими модулями» (рис. 1.9).

Ці початкові модулі використовують пакетну нормалізацію та RMSprop, щоб зменшити кількість параметрів, які потрібно обробити GoogleNet. RMSprop — це алгоритм, який використовує методи адаптивної швидкості навчання.

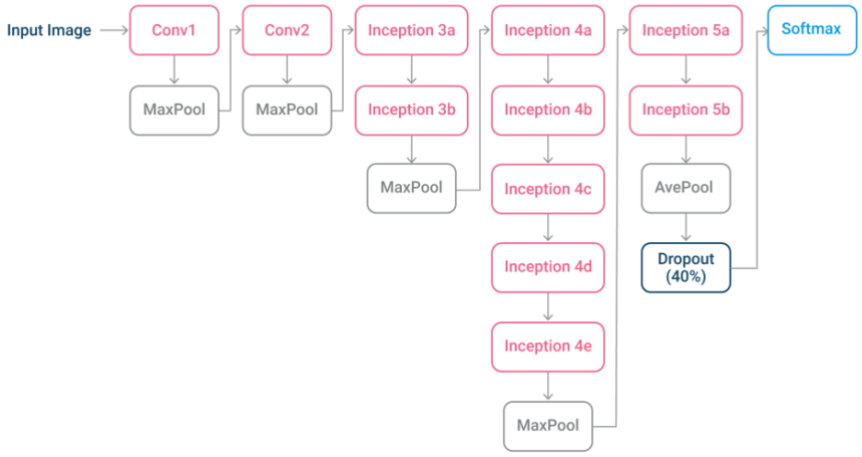


Рис. 1.9. Приклад архітектури нейронної мережі GoogleNet [12]

3. VGGNet (2014) — це 16-рівнева архітектура, деякі варіанти мали 19 рівнів. VGGNet має згорткові шари, шар об'єднання, ще кілька згорткових шарів, шар об'єднання, ще кілька перехідних шарів і так далі. VGG базується на понятті набагато глибшої мережі з меншими фільтрами – вона використовує згортки 3×3 на всьому шляху, що є найменшим розміром фільтра conv, який дивиться лише на деякі сусідні пікселі. Він використовує маленькі фільтри через меншу кількість параметрів, що дозволяє додавати більше шарів. Він має таке ж ефективне сприйнятливе поле, як якщо б у вас був один згортковий шар 7×7 (рис. 1.10).

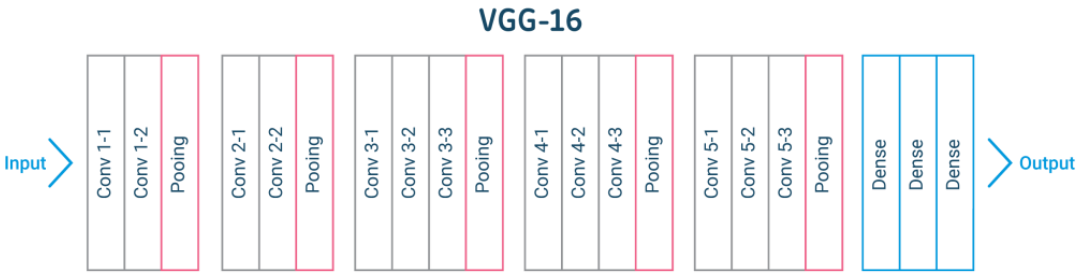


Рис. 1.10. Приклад архітектури нейронної мережі VGGNet [13]

4. ResNet (2015) — це архітектура, яка має велику кількість рівнів. Типово використовувані архітектури варіюються від ResNet-18 з 18 рівнями до ResNet-1202 з 1202 шарами (рис. 1.11). Ці рівні налаштовані з одиниць або «пропуск з'єднань», які дозволяють передавати інформацію на наступні згорточні рівні. ResNet також використовує пакетну нормалізацію для підвищення стабільності мережі. Труднощі в оптимізації малоймовірні, а якщо і будуть, то через викликані зникаючими градієнтами. Це проста мережа навчена так, що забезпечує поширення вперед сигналів мають значення відмінні від нуля дисперсії. Також градієнти зворотного поширення демонструють здорові норми. Таким чином, ні прямі, ні зворотні сигнали не зникають.

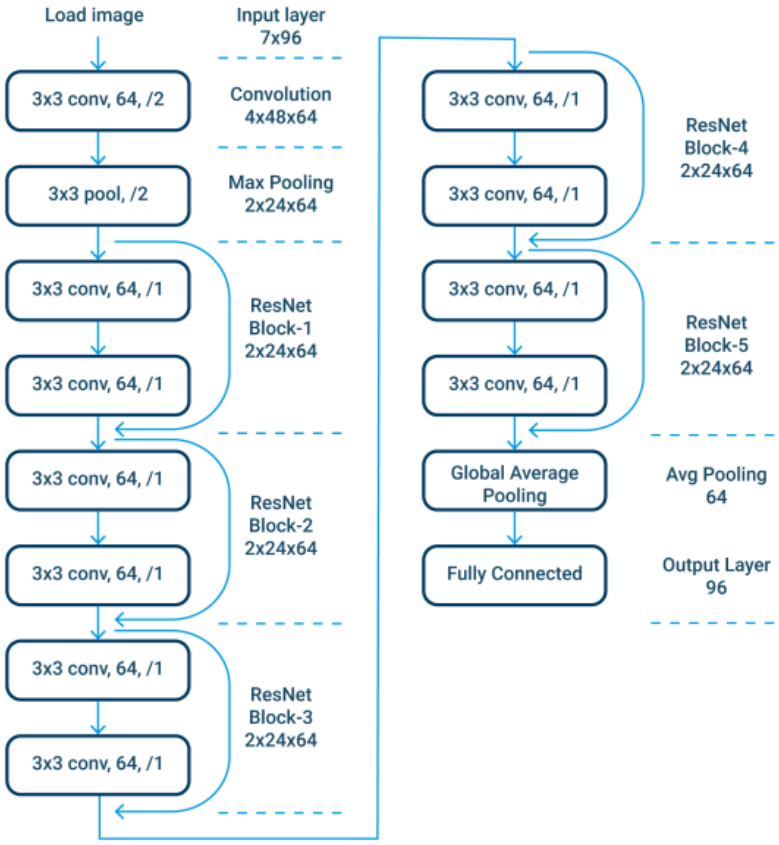


Рис. 1.11. Приклад архітектури нейронної мережі ResNet [14]

5. Xception (2016) — це архітектура, заснована на Inception, яка замінює початкові модулі згортками, які можна розділити по глибині, тобто

згортка по глибині, за якою слідують згортки по точках. Він працює, спочатку фіксуючи кореляцію карт між функціями, а потім просторову кореляцію. Це дозволяє більш ефективно використовувати параметри моделі (рис. 1.12).

Xception розшифровується як «екстремальний початок», він доводить принципи Inception до крайності. У Inception згортки 1x1 використовувалися для стиснення вихідного введення, і для кожного з цих вхідних просторів ми використовували різні типи фільтрів для кожного простору глибини. Xception просто скасовує цей крок. Замість цього він спочатку застосовує фільтри до кожної карти глибини, а потім остаточно стискає вхідний простір за допомогою згортки 1 x 1, застосовуючи її по всій глибині. Цей метод майже ідентичний згортці, що розділяється по глибині, операції, яка використовувалася в розробці нейронної мережі ще в 2014 році. Є ще одна відмінність між Inception і Xception. Наявність або відсутність нелінійності після першої операції. У моделі Inception обидві операції супроводжуються нелінійністю ReLU, однак Xception не створює нелінійності.

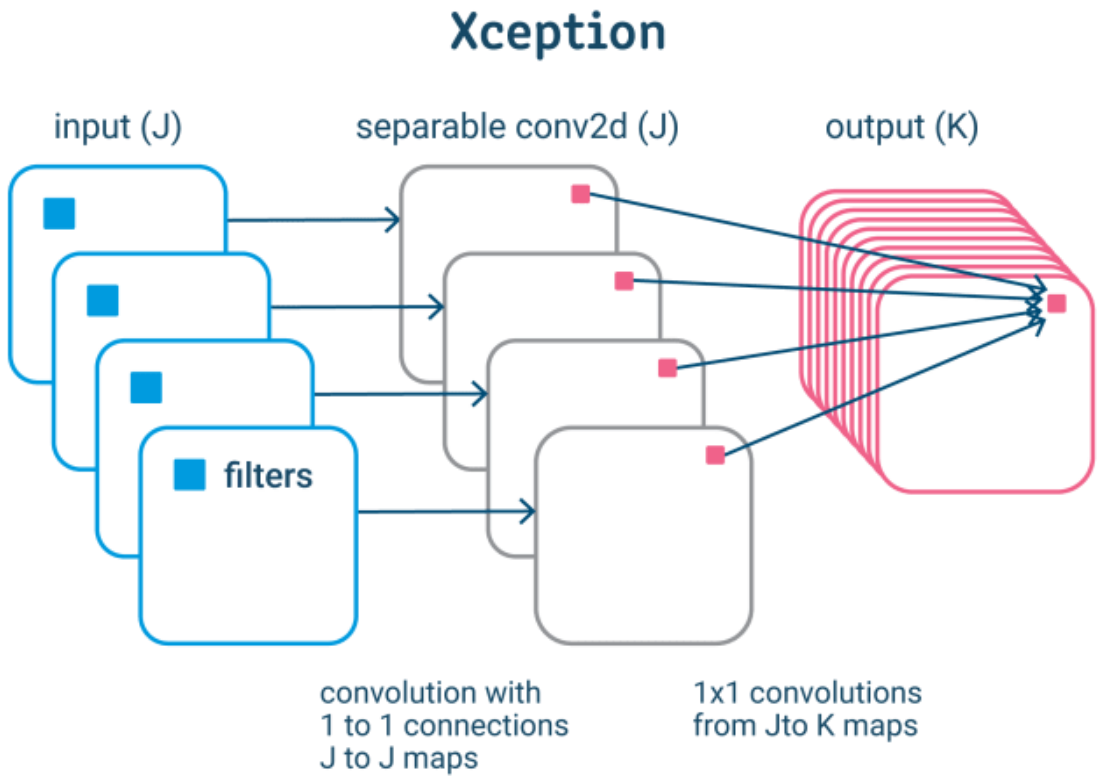


Рис. 1.12. Приклад архітектури нейронної мережі Xception [15]

6. YOLO - детектори об'єктів на основі CNN в основному використовуються для рекомендаційних систем. Моделі YOLO, You Only Look Once – дослівний переклад як «дивишся тільки раз», використовуються для високоефективного виявлення об’єктів. Детектор ділить зображення на систему сіток, і кожна сітка визначає об’єкти всередині неї (рис. 1.13). Їх можна використовувати для виявлення об’єктів у реальному часі на основі потоків даних. Вони вимагають невеликий об’єм обчислювальних ресурсів. Модель має велику історію розвитку, який розпочався ще 2015 році, а до нашого часу дійшла вже сьома версія моделі. Кожна ітерація містить в собі якісні зміни, які покращують роботу моделі відносно минулих версій.

Якщо потрібний швидкий детектор об'єктів, нейромережні моделі сімейства YOLO сьогодні є стандартом де-факто.

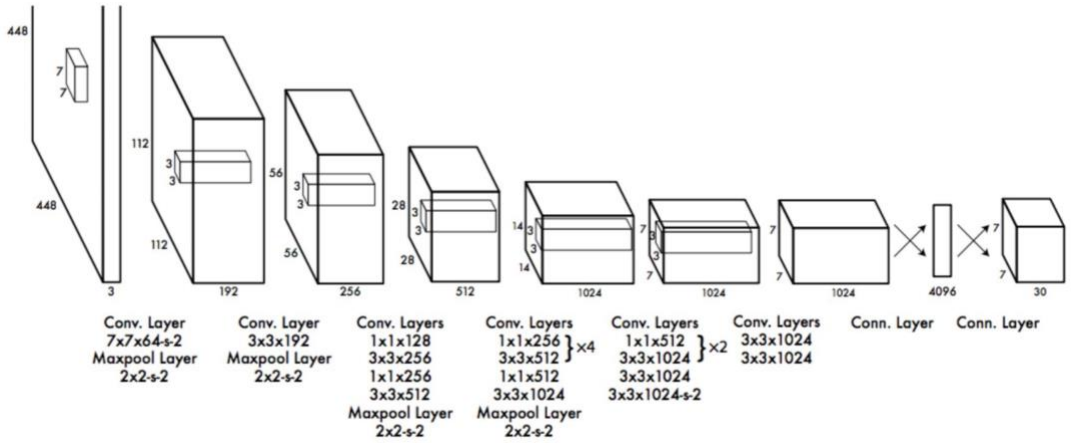


Рис. 1.13. Приклад архітектури нейронної мережі YOLO [16]

Алгоритм YOLO використовує згорткові нейронні мережі для виявлення об’єктів у реальному часі. Як випливає з назви, для виявлення об’єктів алгоритм вимагає лише одного прямого розповсюдження через нейронну мережу. Це означає, що передбачення всього зображення виконується за один прогін алгоритму. CNN використовується для прогнозування різних класів ймовірностей і обмежувальних прямокутників одночасно. Основою для пошуку стає ядро зображення, яке застосовується для реалізації одного прогону алгоритму на зображені.

ВИСНОВКИ ДО РОЗДІЛУ 1

Помітним стає практичне застосування комп’ютерного зору в першу чергу, коли мова йде про ефективність життя. Наприклад, деякі організації використовували технологію комп’ютерного зору під час пандемії, щоб контролювати особовий робочий склад на фактор носіння захисних масок або логістичні компанії можуть покращити обслуговування клієнтів і зменшити витрати, автоматизувавши вимірювання упаковки.

Продуктивність комп’ютерного зору стає все більш важливою, оскільки додатки штучного інтелекту переходять на дрони, мобільні пристрої та транспортні засоби. Крім того, відкалібровані згорткові нейронні мережі покращують можливості виявлення систем машинного зору. Розвиток в цій сфері допомагає впровадженню штучного інтелекту в повсякденних пристроях і мережах інтернету речe, а також продуктивності програм машинного зору.

Всупереч драмі, зображеній у більшості голлівудських фільмів, люди все одно несвідомо використовуватимуть штучний інтелект та його комп’ютерний зір під час виконання повсякденних завдань, наприклад керування транспортним засобом. Інтелектуальні обчислювальні програми також виходять за рамки простої зручності виконання складних завдань, таких як водіння, прогнозування погоди та відновлення балансу нашого пенсійного портфеля.

Застосування штучного інтелекту та комп’ютерного зору водночас є практичним і сюрреалістичним — уже дуже сильно впливаючи на кожен аспект нашого життя, воно продовжуватиме проявляти себе непомітними та дивовижними способами.

РОЗДІЛ 2. РОЗГЛЯД ТА АНАЛІЗ НЕЙРОННОЇ МЕРЕЖІ YOLO

В даний час написано досить велику кількість статей, в яких аналізуються особливості окремих версій YOLO, тому для використання найбільш «корисної» та ефективної моделі для необхідної задачі варто проводити порівняльний аналіз всього сімейства. Ще важливим є розгляд еволюції архітектури, щоб краще зрозуміти, що змінилося, які розробки покращили продуктивність, і, можливо, зробити прогнозування, куди йдуть справи та як буде розвиватись дана модель.

До YOLO основний підхід до виявлення об'єктів зображення полягав у послідовному обході частин вихідного зображення з використанням ковзного вікна різних розмірів, щоб класифікатор міг показати яка частина зображення містить якийсь об'єкт. Підхід є логічним, але дуже повільним [17].

Трохи згодом з'явився спеціальний розділ, який виявляв цікаві місця — прогнозував, де на картинці може бути щось цікаве. Проте цього було мало, бо їх було надто багато, тисячі. Найшвидший із алгоритмів Faster R-CNN обробляв одне зображення в середньому за 0,2 секунди на апаратному рівні, і якщо підрахувати, то це діє 5 кадрів на секунду. Загалом, все було досить сумно, доки з'явився принципово новий підхід.

У попередніх підходах кожен піксель вихідного зображення міг оброблятися нейронною мережею кілька сотень і навіть тисяч разів. І кожного разу ці пікселі проходили через ту саму нейронну мережу, виконуючи одні й ті самі обчислення. Тому швидко набуло популярності питання серед розробників «чи можна щось зробити, щоб уникнути повторення одних і тих самих обчислень?».

Виявляється, це можливо. Але для цього довелося дещо переформулювати завдання. Якщо раніше це була проблема класифікації, то тепер вона стала проблемою регресії.

2.1. YOLOv1

Найперша модель YOLO, також відому як YOLOv1, була розроблена Джозеф Редмон, Сантош Діввала, Росс Гіршик, Алі Фархаді. Перша реалізація була створена на відкритому фреймовку darknet, проте «інтузіасти» та інші розробники зробили власні модифікації на популярних фреймфорках для зручності використання, наприклад: PyTorch, Tenserflow [18].

Системи, які працюють у реальномі часі на Pascal VOC 2007, порівнюють за їх продуктивністю та швидкістю з найпопулярнішими детекторами. Fast YOLO - це найшвидший детектор летких органічних сполук Pascal, також вдвічі точніший, ніж будь-який інший детектор реального часу. Саме YOLO на 10 одиниць показника maP точніше, ніж швидка версія, проте все ж таки має набагато вище швидкість роботи в реальному часі ніж більшість моделей, що можна побачити на порівняльній таблиці (рис. 2.1).

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Рис. 2.1. Порівняння нейронних мереж з YOLO

Конструктивно моделі IOLO складаються з наступних частин:

1) Input - це вхідний шар, який приймає вхідне зображення.

2) Backbone - це частина, в якій вхідне зображення кодується у вигляді ознак.

3) Neck - це додаткові частини моделі, які обробляють закодовані зображення ознак.

4) Head(s) - один або кілька вихідних шарів, які створюють передбачення моделі.

Перша версія мережі заснована на архітектурі GoogLeNet. Це каскад згорткових шарів, перемежованих MaxPool. Каскад закінчився двома повністю з'єднаними шарами.

Також навчили швидшу версію архітектури Fast-YOLO, яка містить менше згорткових шарів, 9 замість 24. Вхідна роздільна здатність обох моделей становила 448 x 448, але основна частина мережі була попередньо навчена як classifier з роздільною здатністю 224 x 224 пікселі.

У цій архітектурі вихідне зображення поділено на комірки P x P, а початкові значення — 7 x 7, і кожна комірка передбачає наявність об'єктів у обмежувальних прямокутниках і рівень впевненості, що будь-який об'єкт присутній у цих комірках, і ймовірність для вказаних класів. Кількість клітинок на кожній стороні непарна, тому в середині зображення є повноцінний піксель. Це має перевагу перед парним числом, тому що знаходжуваний об'єкт часто знаходиться в центрі обраного поля зображення, і в цьому випадку основні проекції робляться на середню комірку. При парній кількості комірок центр може знаходитися, де завгодно між чотирма центральними комірками, в результаті це знижує точнійсть знахоження обмежувальної рамки нейронною мережею. Саме цей факт зменшує довіру до роботи мереж методом обирання парних «пошукових» частин з фотографії.

Confidence показує, наскільки модель впевнена в тому, що цей bounding box містить об'єкт, і наскільки точно, на «думку» мережі, обмежувальна рамка прогнозує його розташування. І архітектурно закладено реалізацію цого

методу пошуку за допомогою добутку ймовірностей того, що знайдений клас присутній. Такий варіант роботи надає використання параметру IoU, який при навчанні порівнює рамку truth та predictions нейронки. Якщо результат зрівняння близький до одиниці, то відповідно об’єкт був знайдений, якщо ж результат близький до 0 - в комірці немає шуканого класу.

Кожен обраний сектор складається з 5 чисел: x, y, w, h і confidence, де (x, y) - координати центру обмежувального боксу в комірці, (w і h) – відповідно ширина і висота bbox відносно розмірів повноцінного зображення, тобто вони нормалізовані і мають значення від 0 до 1. Confidence – це отриманий результат використаної функції отримання IoU між прогнозованим і справжнім bbox. Всі отримані ділянки вхідних даних передбачає умовні ймовірності певного класу об’єктів. Прогнозується лише один набір класів на комірку, незалежно від кількості знайдених блоків.

Таким чином, отримані обмежувальні рамки аналізуються за один крок. Для зменшення опрацювання даних, механізм відсікає всі фрагменти, де значення впевненості нижче певного порогового значення. Цей простий для оптимізації роботи крок має велике значення в кінечному результаті, бо він пришвидшує розпізнавання об’єктів порівняно з конкурентами. В цьому є сенс, тому що ціль реалізації YOLO саме в тому, що всі обмежувальні рамки для кожного зі класів прогнозуються одночасно. Що і можна зрозуміти дивлячись на повну назву нейронної мережі - алгоритм розглядає тип лише один раз.

І хоча точність mAP дещо знизилася порівняно з попередніми алгоритмами, аналіз у реальному часі важливіший для проблем комп’ютерного зору.

Оскільки комірки, прилеглі до центру об'єкта, також можуть генерувати bboxs, що призводить до їх надлишку, необхідно вибрати найкращу. Для цього використовується техніка немаксимального придушення, яка працює наступним чином. Усі bboxs в цій категорії взяті з фото. Ті, чия довіра нижче певного порогу, будуть відхилені. В іншому випадку виконується процедура

попарного порівняння відповідно до IoU. Якщо IoU > 0,5 для двох порівнюваних, то bbox з нижчим коефіцієнтом confidence відхиляється. В іншому випадку обидві сторони залишаються в списку. Так розрвіджуються подібний мітки об’єктів.

Функція втрат є складною і має цікавий вигляд, вона представлена на рис. 2.2.

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

Рис. 2.2. Функція втрат в YOLO

Перший доданок – це втрати координат центру об'єкта, другий - розміри обмежувальної рамки, третій - клас об'єкта, четвертий – це клас, якщо об'єкт відсутній, п'ятий - ймовірність знайти об'єкт в bbox.

Лямбда-коефіцієнти необхідні, щоб confidence не впав до нуля, оскільки більшість комірок не мають об’єктів. **1(obj, i)** означає, що центр об’єкта з’являється в клітинці і **1(obj, i, j)** означає, що j-те поле b клітинки і відповідає за це передбачення.

Переваги першої версії YOLO:

- висока швидкість роботи детектора по зрівнянню з конкурентами, що можна бачити на представлених рисунках вище;

- краща можливість узагальнення, ніж у тогочасних конкурентів, тести в показали кращу продуктивність на опрацювання зображень, а саме тренування проводилося на славнозвісному датасеті ImageNet;
- менше помилкових спрацювань на задньому фоні зображення.

Обмеження в роботі моделі:

- обмежено двома розміченими рамками та одним класом об’єктів на клітинку, а саме означає, що деякі дрібні предмети менш помітні та надають більше помилок у знаходженні об’єктів;
- кілька послідовних дискретизації вихідного зображення призводять до погіршення точності;
- loss призначений для покарання за помилки як на великих, так і на малих об’єктах, була спроба компенсувати цей ефект, вилучивши квадратний корінь з величини, але це не повністю усунуло проблему.

2.2. YOLOv2/YOLO9000

Наступною ітерацію нейронної мережі була її нова версію YOLOv2, або ще її другою назвою YOLO9000. Реліз моделі відбувся за значного вкладу Джозефа Редмона та Алі Фархаді. Після чого незалежні розробники створили інфраструктуру для навчання даної нейронки на популярних фреймворках таких як: open-source project Darknet, Keras, Tenser-Flow та PyTorch [19].

Покращена версія YOLO була протестована на загально-доступному датасеті Pascal VOC 2007 та Pascal VOC 2012. Результати порівняння можна розглянути на наступному зображені (рис. 2.3). Кожен з моделей pre-trained на розпізнавальному датасеті Pascal 2007 VOC. І YOLOv2 є найшвидшим і точнішим, ніж інші методи виявлення. Модель також може працювати з різною роздільною здатністю, забезпечуючи легкий баланс між швидкістю та точністю. Кожен запис YOLOv2 — це, по суті, та сама модель, навчена з тими самими вагами, тільки в іншому масштабі вхідних зображень до якого зводиться кожен екземпляр набору.

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

Рис. 2.3. Порівняння продуктивності YOLOv2

Архітектурні особливості, що були привнесені в нове покоління нейронних мереж YOLOv2, розробники внесли ряд поліпшень у першу версію моделі, а саме:

1) Було видалено dropout та додано batchnorm до всіх шарів плетіння.

2) Попередньо навчений, як класифікатор з роздільною здатністю 448 x 448, YOLOv1 при 224 x 224, остаточну сітку потім масштабували до вхідних даних 416 x 416, щоб отримати непарну кількість 13 x 13 комірок для покращення знаходження точних кордонів рамок знайдених об’єктів.

3) Повністю об’єднані шари видалено, натомість за основу взято використовувати якорів повної згортки для прогнозування блоків, за схожим алгоритмом роботи як у Faster RCNN, що у результаті допомогло менше втрачати просторової інформації, як це було у першій версії з повним з’єднанням шарів.

4) Видалено один maxpool, щоб збільшити деталізацію, тобто роздільну здатність функції, коли у v1 було лише 98 bboxs на кадр, то з якорями v2 в результаті отримується понад 1000 рамок, тоді як mAP трохи впав, але recall значно зріс, що дозволило підвищити загальну точність.

5) Змінили попередні вимірювання, а саме вимірювання розмірів та позиції блоків, які рандомізуються не вручну, як у FasterRCNN, а автоматично за допомогою кешування k-means. Хоча похибка виявлення була вищою для стандартних k-середніх з евклідовою відстанню в малих bboxs, для k-середніх було обрано іншу міру відстані, а саме 1 – IoU, значення рамки мінус центроїд. Значення 5 було обрано як компромісний варіант за кількістю кластерів, тому що випробування показали, що для п’яти фокусних точок, обраних таким чином, що середній IoU був більш-менш таким же, як і для дев’яти якорів.

6) Важливою зміною було пряме передбачення розташування. З якорями спочатку існувала нестабільність у створенні сітки, пов’язаної із зазначенням центральних координат (x, y), оскільки ваги сітки були ініційовані випадковим чином, а прогнозування координат було лінійним із необмеженими коефіцієнтами. Замість прогнозування зміщення від опорного центру, коли діапазон дійсного коефіцієнта знаходиться в інтервалі [-1; 1], вирішено передбачити bbox відносно центру комірки, для коефіцієнтів з інтервалом [0; 1] і використовується sigmoid, щоб обмежити його. Мережа надає п’ять знайдених зразків на комірку, де кожен з них має п’ять значень: tx, ty, tw, th, to. Прогнозовані параметри обчислюються таким чином (рис. 2.4):

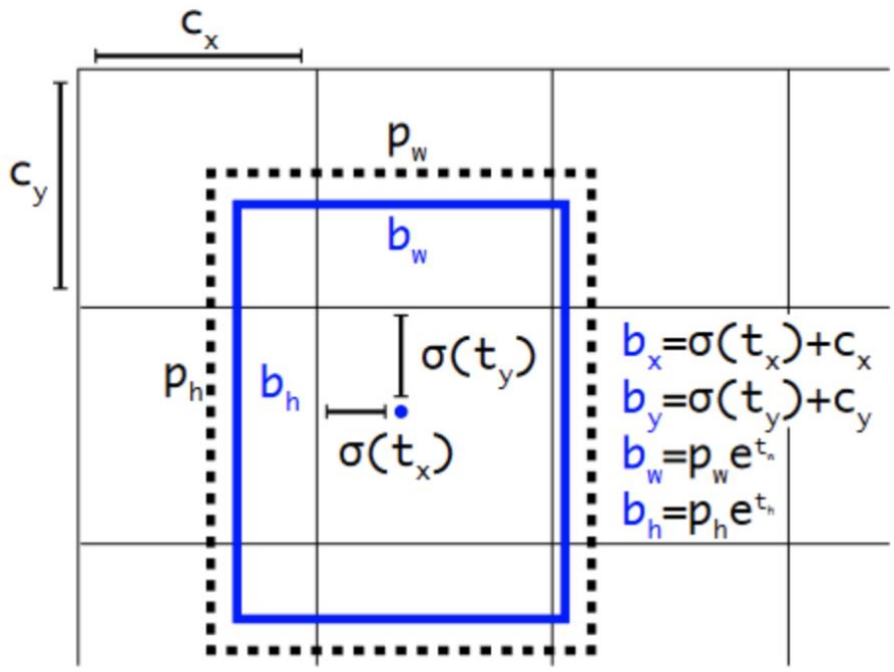


Рис. 2.4. Наглядний приклад обчислення параметрів в YOLOv2

7) Обмежувальна рамка з попередніми вимірюваннями і прогнозованим положенням. Модель передбачає ширину та висоту коробки як зміщення від центрів кластера, а саме відбувається прогнозування координат центру обмежувальної рамки відносно місця застосування фільтра за допомогою сигмоїдальної функції.

$$b_x = \sigma(t_x) + c_x \quad (2.1)$$

$$b_y = \sigma(t_y) + c_y \quad (2.2)$$

$$b_w = p_w e^{t_w} \quad (2.3)$$

$$b_h = p_h e^{t_h} \quad (2.4)$$

$$Pr(object) * IOU(b, object) = \sigma(t_o). \quad (2.5)$$

8) Властивості «дрібного зерна», тобто розбиває велику за розмірами матрицю зображення на сегменти. Карта об'єктів тепер має розміри 13x13.

9) Додано багатомасштабне навчання. Оскільки мережа повністю є згортковою, її роздільну здатність можна змінити на льоту, просто змінивши роздільну здатність вхідного зображення. Для підвищення надійності мережі її вхідна роздільна здатність змінювалася кожні 10 пакетів отриманих даних. Якщо розмір сітки зменшується в 32 рази, вхідна роздільна здатність вибирається з набору {320, 352, ..., 608}. Розмір сітки було змінено з 320x320 на 608x608 і навчання продовжується.

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

Рис. 2.5. Результуюча архітектура YOLOv2

10) Прискорення. VGG-16, який вважається основою першої версії моделі, був надто важким, тому друга версія використовувала Darknet-19 замість нього. Після навчання класифікатора був видалений останній згортковий шар мережі, додано три згорткові шари розміром 3x3 з 1024 фільтрами і останній 1x1 з кількістю виходів, необхідних для виявлення. У VOC було п’ять bboxs кожен з п’ятьма координатами та двадцятьма класами обмежувальних рамок, загалом 125 фільтрів (рис. 2.5).

11) Ієрархічна класифікація. У той час як у першій версії класи належали до однієї категорії об’єктів і були взаємовиключними, у другій версії була представлена структура дерева WordNet, яка є орієнтованим графом. Класи в кожній категорії є взаємовиключними та мають власний softmax. Наприклад, якщо зображення показує собаку з мережі відомої породи, мережа поверне клас як для собаки, так і для конкретної породи. Якщо це собака невідомої породи, повертається лише клас собаки. Цікавим прикладом буде розподілення на групи класи відомого загальнодоступного датасету для класифікації ImageNet, який має в загальному тисячу класів, де моделі, які попередньо навчені на цьому наборі даних, мають загальну функцію softmax. Використовуючи WordTree модель виконує кілька операцій softmax над кожною групою назв класів (рис. 2.6).

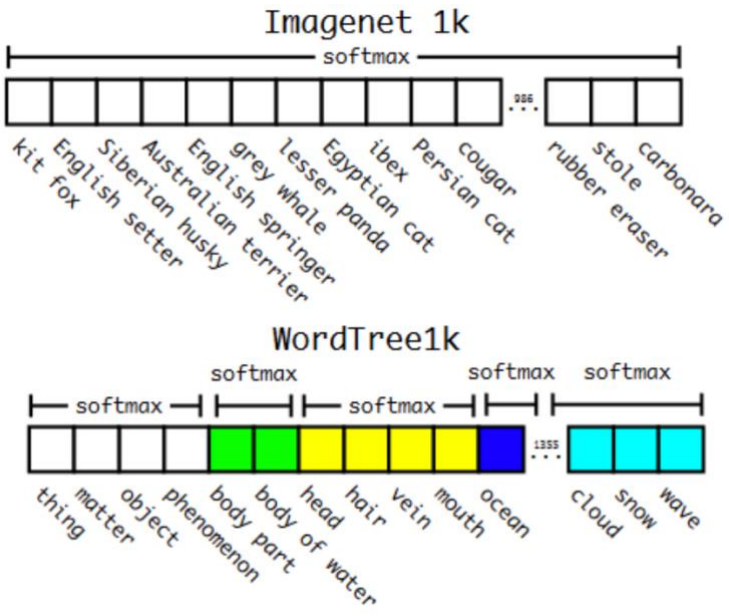


Рис. 2.6. Приклад використання WordTree в YOLOv2 на датасеті ImageNet

В загальному YOLOv2 має кращу архітектурну складову та надає на виході точніші результати пошуку об’єктів. Покращення в цій версії нейронної мережі призвели до лідерства нейронки не тільки по швидкості, а й по параметру mAP, а також до знаходження об’єктів малого розміру.

2.3. YOLOv3

Джозеф Редмон та Алі Фахарді продовжили оптимізувати створену ними модель комп’ютерного зору, покращуючи її можливості в знаходженні об’єктів у реальному часі з чіткою детекцією та швидким опрацюванням потокової інформації. Третя версію нейронки була представлена у 2018 році з очікуваною назвою YOLOv3. Для вимірювання якості роботи мережі, було проведено тестування з попередньо навченими конкурентами, які також використовують для знаходження об’єктів в computer vision [21].

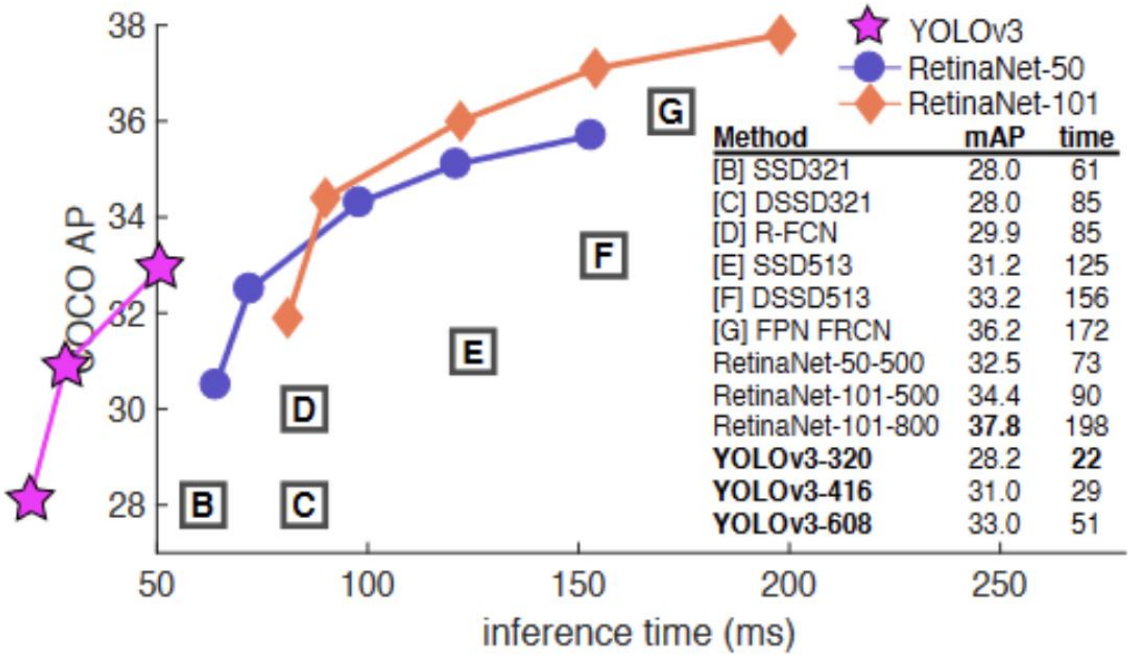


Рис. 2.7. Порівняння продуктивності роботи YOLOv3 з конкуруючими моделями

Архітектурними «фічами» даної версії є покращення набору опрацьовуючих дані алгоритмів без значних кардинальних змін, проте це означає нове покоління нейронної мережі залишилось без покращень у роботі.

Покращення полягали оцінки об’єктивності, тобто ймовірність того, що об’єкт знаходиться в певній знайденій рамці, тепер також обчислюється за допомогою сигмоїди для кожного детекта.

Архітектурно нейронна мережа перейшла від класифікації з кількома класами до класифікації з кількома мітками, а саме це стало можливо за умови, що потрібно було відмовилися від softmax на користь бінарної крос-ентропії, яка в даному випадку давала приріст в якості обробки отримануваної інформації.

Також змінилось прогнозування обмежувальної рамки, яке зроблене для bbox за трьома шкалами, розмір вихідного тензора:

$$N * N * \left(3 * \left(4 + 1 + \text{number}_{\text{classes}}\right)\right) \qquad (2.6)$$

де $number\ classes -$ кількість опрацьовуваних нейронкою класів

Розробники перерахували пріоритети за допомогою k-середніх і отримали дев’ять полів на трьох шкалах. YOLOv3 отримала нову, глибшу та точнішу функцію екстрактора колонок Darknet-53 за основою та особливістю знайденою нейронкою (рис. 2.8).

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Рис. 2.8. Відображення зв’язних шарів нейронної мережі YOLOv3

На виході реліз нейронної мережі YOLOv3 за якістю точного детекту можна порівняти з ResNet-152, що можна бачити на представленому рис. 2.7. вище, проте мережа You Only Look Once вимагає в 1.5 рази менше ресурсів через меншу кількість операцій по знаходженню об’єктів та в post-processing обробці отриманої інформації, що на виході дає нам в два рази, а інколи і більше, FPS за рахунок кращого оперування з графічним процесором машини (рис. 2.9).

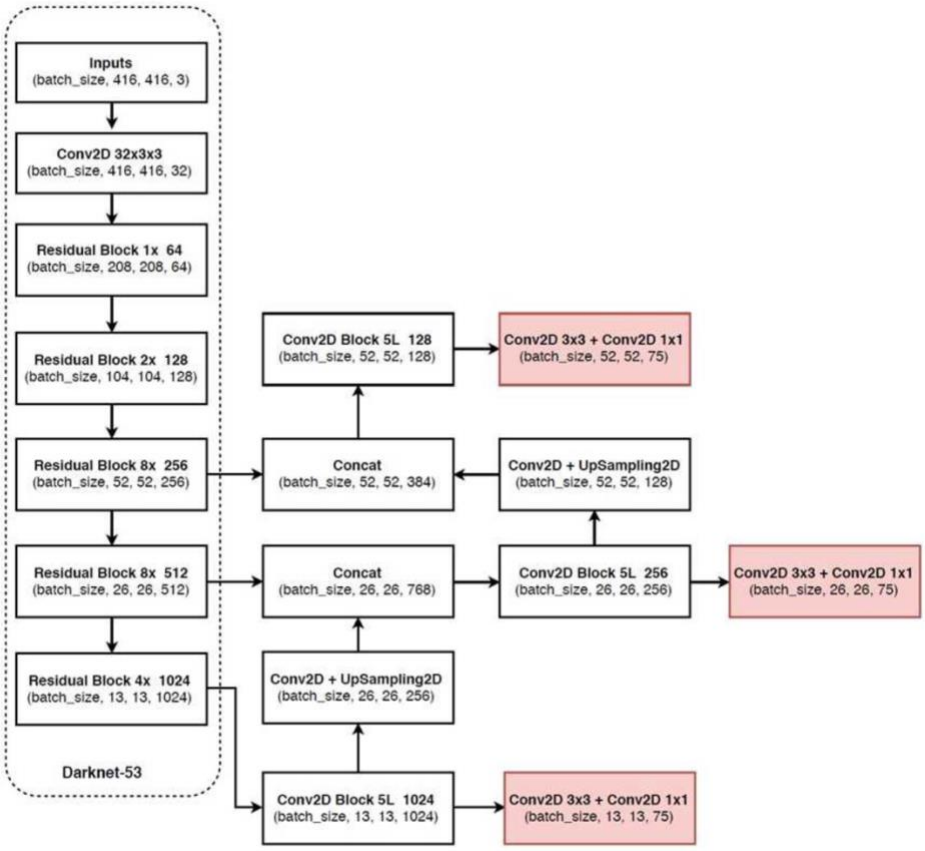


Рис. 2.9. Фінальна архітектура YOLOv3

Проте не всі оновлення в роботі мережі дали позитивні показники у роботі. Обмежувальні рамки знайдених об’єктів координує передбачення руху за допомогою лінійної, а не логістичної активації, що більше сповільнює роботу нейронки і не дає максимальної ефективності. Також виникла проблема локального мінімуму функції та затухання градєнту через focal loss, що вплинуло на значення параметра mAP та зменшило цей показник на дві одиниці в абсолютному значенні.

Цікавим ще підходом, що не спрацював, є подвійний поріг перевірки параметру IOU для визначення правдивості. У Faster R-CNN є два пороги для IOU, які визначають позитивний або негативний, а саме значення [0,7; 1] позитивний, (0.3; 0.7) ігнорований, [0; 0,3] негативний.

Перевагами YOLOv3 перед конкурента в точності та рівні виявлення об’єктів на момент запуску.

2.4. YOLOv4/Scaled YOLOv4

У 2020 році було представлене продовження роботи над YOLO, проте склад розробників змінився на: Олексій Бочковський, Чиен-Яо Ван, Хун-Юань Марк Ляо. Джозеф Редмон вирішив завершити свою роботу у розвитку цього open-source проекту з відкрити кодом. Насправді, спочатку була випущена версія YOLOv4, а трішки пізніше був реліз масштабованої версії Scaled YOLOv4 [22].

Значною зміною в цій версії був датасет для попереднього навчання, а саме MS COCO Object Detection, який був представлений компанією Microsoft. Це новий у своєму роді набір даних, що мав цікавий підхід до формування датасету. Він містить 91 клас, де кожен з яких максимально репрезентується у відповідній до природного контексту об’єкта. Такий підхід направлений на те, щоб допомогти нейронній мережі при навчанні створити гіпотези та характеристики щодо класу в його звичному середовищі. Саме цей підхід покращує роботу моделі у знаходженні об’єктів у реальному часі. Датасет сформований так, щоб його могла зрозуміти навіть дитина [23].

Розглядаючи вміст даного набору даних, то можна відмітити два з половиною мільйони позначень на 328 тисячах різних зображень. Всі фото мають складну сцену відображення звичних речей у природному їм контексті. Розмічений датасет у декількох форматах, а саме з обмежувальними рамками для об’єтів та сегментуючи окремі частини, що відповідають потрібному класу. Приорітетним в складанні цього набору даних було показати звичні для

людини об’єкти під різним кутом та поглядом, щоб нейронна мережа змогла максимально відтворити «погляд» людського ока.

Використання MC COCO Object Detection у тренуванні допомагає закласти перші гіпотези у «баченні» нейронної мережі. Зрівнюючи YOLOv4 з конкурентно спроможними моделями, такими як EfficientDet, ASFF та попередньою версією YOLOv3. З наведеного нижче графіка на рис. 2.10, можна відмітити якість роботи всіх тестувальних мереж, а саме те, що YOLOv4 відпрацьовує вдвічі швидше ніж EfficientDet і ASFF із порівнянною продуктивністю. Також варто відмітити, що нова версію моделі покращує показники точності зі збільшення FPS обробки потоку інформаційних даних на 10-12% відносно старішої версії YOLOv3.

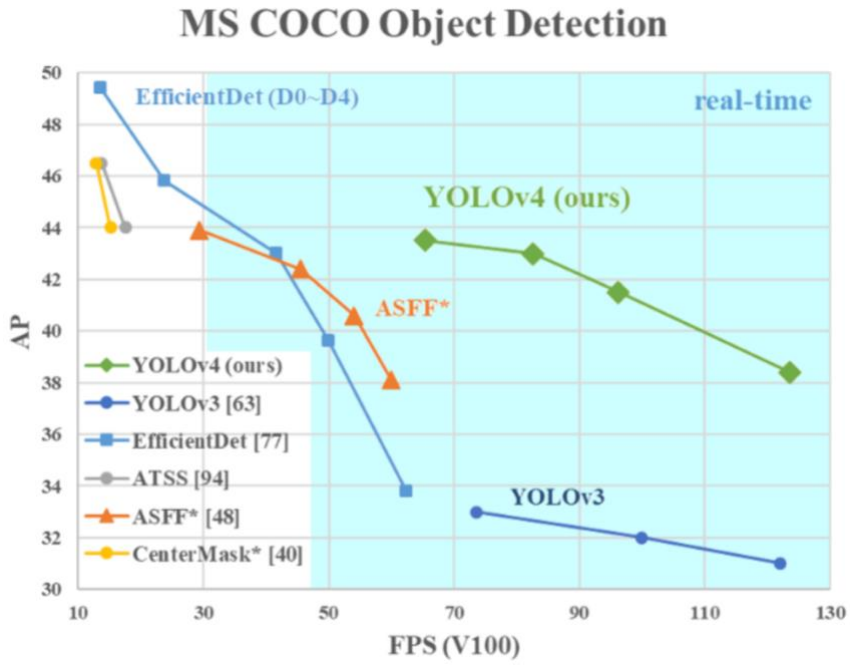


Рис. 2.10. Порівняння нейронних мереж з YOLOv4

Четверта версія використовує більш ефективний CSPDarknet53 в своїй основі, ніж версія 3. CSP вказує на наявність часткових перехресних з’єднань, типу з’єднання між несуміжними рівнями мережі. При цьому кількість шарів залишилася незмінними. Також додано модуль SPP. Детальніше модуль CSPDarknet53 можна розглянути на рис. 2.11.

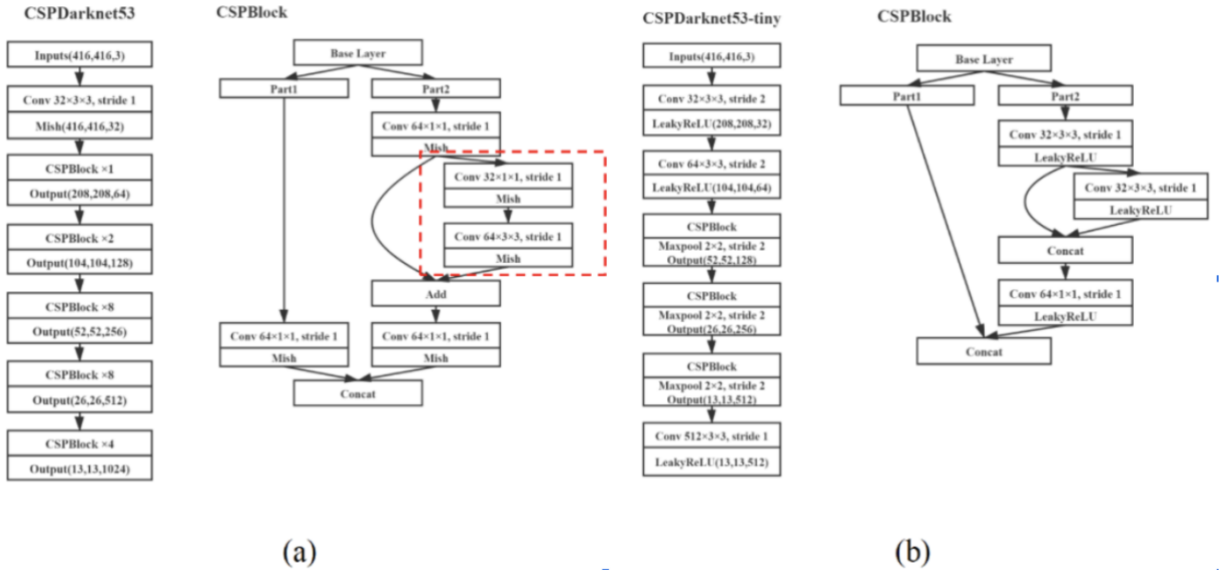


Рис. 2.11. Структура CSPDarknet53 (a) і CSPDarknet53-tini (b)

Так звана «шия» в архітектурі нейронної мережі складається з модулів PANet. Натомість блок FPN використовується для агрегації маршрутів, а саме для конкатенації, замість підсумовування, активації в різних масштабах. А сам вихідний шар, як його ще називають головою або керівником мережі, не змінився та має ті ж якорі для тренування.

Крім архітектурних змін, деякі покращення були внесені в процес навчання.

Застосований двоетапний метод доповнення SAT (Self-Adversarial Training). У першій фазі замість ваги мережі змінюється зображення до такого стану, що для нейронки воно виглядає так, ніби потрібний предмет відсутній, це ще називають змагальною атакою. На другому кроці мережа навчається розпізнавати об’єкт на зображенні, що було змінено відносно першого кроку.

Розширилось рецептивне поле і задіяння механізму уваги. З’явилось безліч додаткових видів аугментації та балансування всіх доступних класів.

Аугментація зображень створює нові навчальні приклади з наявних навчальних даних. Неможливо по-справжньому зафіксувати зображення для кожного сценарію реального світу, який може бути поставлено перед моделлю. Таким чином, коригування наявних навчальних даних для

узагальнення на інші ситуації дозволяє моделі вчитися на більш широкому спектрі ситуацій.

Є ряд різноманітних видів аугментації, що активно використовуються в комп'ютерному зорі. На справді, цей інструмент для покращення роботи нейронки має дуже широкий спектр можливостей, тому що містить в собі багато різновидів зміни вхідного зображення в залежності від необхідного результату. Таким чином є делька груп, які поділяють аугментацію, а саме на спотворення та оклюзію зображення.

Спотворення зображень відбувається за допомогою накладання фільтрів на вхідне зображення, що на виході дає «не точне» відображення об'єктів, що «мотивує» нейронку побудувати нові залежності та гіпотези пошуку. Саме ж спотворення даних також ділиться на два види:

- фонометричне спотворення – це фільтри для зміни яскравості, контрастності, насиченості та шуму зображення;
- геометричне спотворення – це зміни фото, як можна зрозуміти з назви, які пов'язані з інтерпретації вхідної інформації, тобто зображення, в різних геометричних формах: довільне масштабування, обрізання, гортання та відзеркалення зображення.

Обидва ці методи вносили зміни в піксельну структуру в матричне представлення зображення, тому, виконуючи незамисловаті кроки, вхідне фото можна відновити.

Оклюзія зображень включає в собі набір фільтрів, які вносять безпосередні зміни у вхідне фото.

Можна виділити такі основні фільтри:

1) Довільне вирізання – це техніка зміни зображення, яка замінює зображення на випадково обраними значення або середнім значенням пікселя навчального набору.

2) Виріз — квадратні області маскуються під час навчання. Області вирізу приховані лише від першого шару CNN. Це дуже схоже на випадкове стирання, але зі постійним значенням у накладеній оклюзії.

3) «Хованки» - для цього методу необхідно розділити зображення на сітку S x S, далі приховати кожен сектор з певною ймовірністю, що дозволяє моделі дізнатися, як виглядає об’єкт, не вивчаючи лише те, як виглядає окрема частина об’єкта.

4) Маска сітки — регіони зображення приховані у формі сітки, подібно до «хованок», це змушує нашу модель вивчати складові частини того, з чого складається окремий об’єкт.

5) Змішування - опукле накладення пар зображень та їх міток.

6) Згорточна нейронна мережа використовує в своїй основі backbone, що об’єднує та формує характеристики зображення з різною деталізацією.

Покращення роботи основи YOLOv4 можна досягти за допомогою таких аугментацій даних :

- для навчання: CutMix + Mosaic augmentation, DropBlock регулярізація, Classmark smoothing;
- для валідації: активація Mish, часткові перехресні з’єднання (CSP), зважені залишкові множинні вхідні з’єднання (MiVRC).

Для покращення роботи детектора, тобто вихідного слою нейронної мережі, також є набір дій, що допоможуть максимально збільшити показники знаходження об’єктів [24].

Для навчання краще використовувати: CIoU-loss, CmNN, DropBlock, Mosaic, SAT, видалити чутливість мережі, зміни в опорних якорях для наземної реальності, планувальник косинусного вирівнюваннння, оптимізація гіперпараметрів, рандомна змені часу тренування.

А для покращення результатів валідації варто застосувати: Mish, блок SPP, що є просторовою пірамідальною асоціацією, блок SAM, що є модулем просторової уваги, PAN, DIoU-NMS.

Аугментації зображень для навчання нейронної мережі, які не впливають на FPS, але покращують точність можна розглянути на рис. 2.12.

Рис. 2.12. Види аугментації для покращення роботи нейронки

Через півроку після публікації першої статті v4 була опублікувана ще одна, в якій відображено механізм масштабування мережевої архітектури. Цей механізм включає не тільки масштабування вхідної роздільної здатності, ширини та глибини таблиці, але й масштабування самої структури таблиці.

Модуль YOLOv4 не тільки швидший і точніший за конкурентів, його також можна тренувати на відносно слабкому апаратному забезпеченні.Також у четверту версію вбудовано в OpenCV, тому ви можете отримати до нього прямий доступ без даркнету

2.5. YOLOv5

Наступним, хто привніс покражещення в YOLO, був Глен Джохер, проте цей розробник не брав участі в розробці архітектури попередніх версій, а лише в реалізації, тому законність використання назви "YOLOv5" виглядає етично сумнівною. Про це було багато розмов в Інтернеті, але поки що назва прижилася. Перше на що варто глянути, це порівняння продуктивності нової версії з попередніми версіями YOLO (рис. 2.13). Архітектурно, якщо глянути на всі шари згорткової нейронної мережі, то нова версія – це еволюція YOLOv3, а не YOLOv4 і з’явилась вона через декілька місяців після релізу

четвертої версії. З графіку порівняння вище можна відмітити, що продуктивність у v5 краща, ніж у v3, проте гірша, ніж у v4.

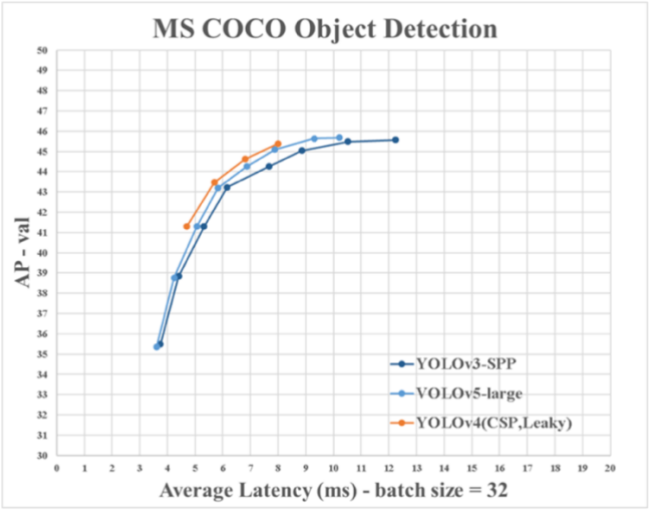


Рис. 2.13. Порівняння продуктивності YOLOv5 з попередніми версіями

Архітектура мережі YOLOv5 складається з трьох частин:

1) Основа - CSPDarknet.
2) Детектор - PANeti.
3) Вихідний шар Yolo.

Дані спочатку надсилаються до CSPDarknet для вилучення функцій, а потім надсилаються до PANet для об'єднання функцій. Нарешті, рівень YOLO надає результати розпізнавання класу, оцінки, розташування та розміру. Для покращення роботи даної моделі нейронної мережі варто застосовувати аугментацію у вигляді масштабування, налаштування колірного простору та мозаїки.

Реалізація п'ятої версії таке ж, як у чертвертої, також використовується вузьке місце функціональності CSP та PANet для агрегування функцій.

Головною перевагою YOLOv5 – це добре розроблене сховище, яке можна розгорнути на мобільних і енергозберігаючих пристроях та швидке навчання. Саме тому ця версія є популярною у розроблених девайсах Internet of things, тобто інтернет речей, що уособлює в собі новітні розробки по автоматизації багатьох сфер людського життя.

ВИСНОВКИ ДО РОЗДІЛУ 2

Виявлення об’єктів — це велика область комп’ютерного зору та одне з найважливіших застосувань в реальному світі. З одного боку, його можна використовувати для побудови автономних систем, які переміщують агентів у середовищі – будь то роботи, що виконують завдання, або самокеровані автомобілі, але це вимагає перетину з іншими сферами. Однак виявлення аномалій, наприклад, бракованих продуктів на лінії, визначення місцезнаходження об’єктів на зображеннях, розпізнавання обличчя та інші застосування виявлення об’єктів можна виконати без перетину з іншими сферами діяльності.

Виявлення об’єктів не є настільки стандартизованим, як класифікація зображень, головним чином тому, що більшість нових розробок, як правило, виконуються окремими дослідниками, супроводжувачами та розробниками.

Це робить виявлення об’єктів дещо складнішим, зазвичай більш детальним і менш доступним, ніж класифікація зображень. Одна з головних переваг перебування в екосистемі полягає в тому, що вона дає можливість не шукати корисну інформацію про хороші практики, інструменти та підходи до використання. З виявленням об’єктів – більшості людей доводиться набагато більше досліджувати ландшафт поля, щоб отримати хороший контроль.

YOLO (You Only Look Once) — це методологія, а також сімейство моделей, створених для виявлення об’єктів. З моменту створення в 2015 році YOLOv1, YOLOv2 (YOLO9000) і YOLOv3 були запропоновані тим самим автором і спільнота глибокого навчання продовжувала вдосконалювати відкритий вихідний код протягом наступних років [26].

Проект абстрагує непотрібні деталі, водночас дозволяє налаштовувати практично всі формати експорту і використовує дивовижні методи, які роблять увесь проект ефективним і оптимальним, наскільки це можливо. Справді, це приклад краси впровадження програмного забезпечення з відкритим кодом і того, як воно впливає на світ, у якому ми живемо.

РОЗДІЛ 3. АНАЛІЗ ВИКОРИСТОВУВАНИХ ТЕХНОЛОГІЙ

3.1. Data Engineering – основні положення та використання.

3.1.1. Формування розуміння, що таке Data Engineering та сферу його діяльності.

Інженери даних – це ті люди, які потрібні для виконання завдань пов’язаних із розробкою даних. Вони, як правило, працюють у сфері IT або комп’ютерних наук, допомагають керувати необробленими даними, щоб зробити їх доступними та придатними для використання. Інженери даних знають, як створювати цінність із необроблених і неструктурованих даних, і транспортувати їх з одного місця в інше без будь-яких змін [27].

Ландшафт технічної кар’єри постійно змінюється, а розвиток роботи в області штучного інтелекту відкрив нові можливості, які відрізняються від традиційних робіт з розробки даних і науки про дані. Інженери з даних створюють канали даних та інфраструктуру, щоб гарантувати, що перетворені дані завжди доступні. Аналітики даних аналізують і моделюють ці дані, щоб розробити нові функції продукту або покращити бізнес-результати. Метою створення команд ШІ є створення інтелектуальних систем, які зосереджені на дуже конкретних завданнях і які можна інтегрувати з масштабованими перетвореннями даних у роботі з інженерією даних і продуктами даних, а також бізнес-рішеннями в роботі з наукою про дані.

Відмінності між сферами діяльності штучного інтелекту, наукою про дані та розробкою даних можуть значно відрізнятися від компанії до компанії та команди до команди.

Обсяг даних, які компанії збирають і обмінюються, збільшується кожного дня. Щоб увімкнути функціональність пристроїв і продуктів обробки даних, інженери повинні розробити надійні та масштабовані архітектури створення та зберігання даних. Інженери, які будують ці системи, повинні ретельно враховувати поточні та майбутні потреби компанії.

Таким чином, основна повсякденна діяльність інженера з даних включає створення систем даних для вирішення нових завдань проектування, покращення та підтримку існуючих архітектур, інтеграцію систем з новими або кращими інструментами та синхронізацію з членами команди для забезпечення високої якості роботи та потоку продуктів.

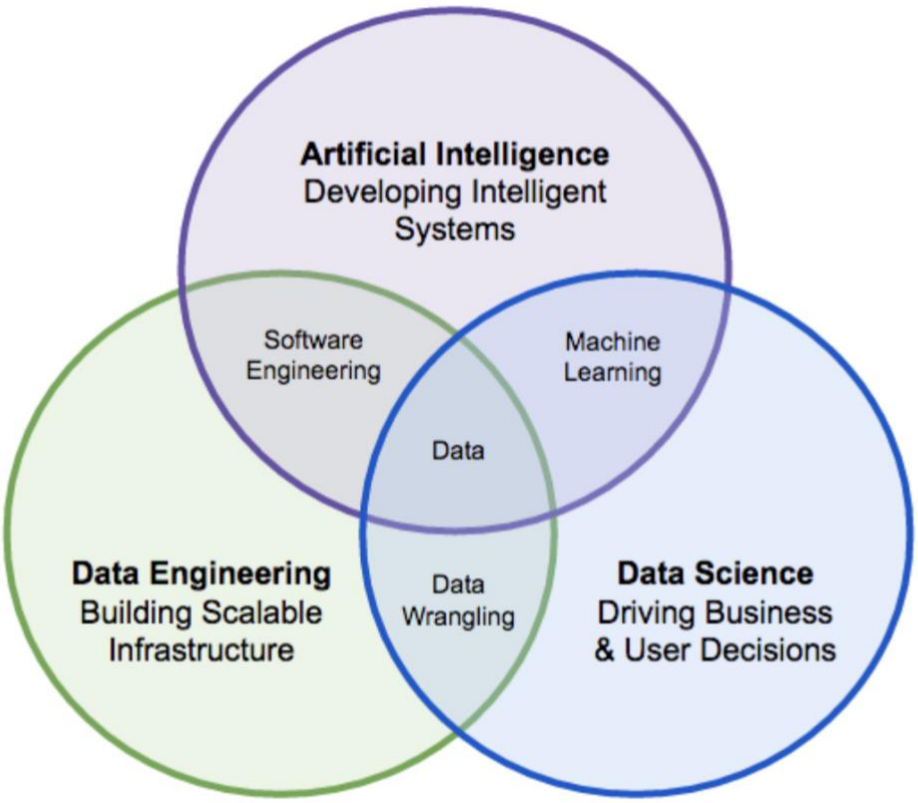


Рис. 3.1. Відображення необхідність даних для всіх ШІ, які з ними працюють

Різниця між конкретними ролями часто нечітка. Багато ролей у сфері обробки даних часто вимагають знання сучасних інструментів розробки даних, щоб отримати потрібні дані, у той час як дедалі більше ролей у галузі обробки даних виконують критичний аналіз і впроваджують машинне навчання у свої процеси (рис. 3.1). Існує також значне збігання між роботою

спеціалістів зі штучного інтелекту та науковців та інженерів обробки даних. Фундаментальні ідеї та знання, які лежать в основі аналізу даних, підтримують навички, необхідні для створення моделей ШІ. Для цих моделей зазвичай потрібні дуже великі набори даних. Таким чином, незважаючи на те, що ефективне маніпулювання та використання великих обсягів даних є фундаментальним аспектом роботи з розробки даних, воно має важливе значення для передових систем ШІ.

Фундаментальна відмінність між ШІ та Data Science/Engineering полягає в природі основної мети ШІ: створювати інтелектуальні системи, які генерують власні функції та знання домену, і забезпечують ефективність завдань, які є близькими, на або вище людського експертного рівня. [28] Для досягнення цієї мети, природно, потрібен інший та додатковий набір навичок, зосереджених на моделях і техніках, які поєднують найсучасніші дослідження та практику ШІ.

3.1.2. Сфери комп'ютерного зору, де застосовуються великі набори даних.

Комп'ютерний зір — це область штучного інтелекту, яка дозволяє комп'ютерам отримувати значущу інформацію та дані з цифрових зображень, відео та інших візуальних даних. Computer Vision використовувалося десятиліттями та набуло популярності в 2012 році з випуском AlexNet, архітектури згорткової нейронної мереж, яка виграла конкурс ImageNet на широкомасштабне візуальне розпізнавання [29]. Успіх AlexNet започаткував багато кращих практик для CNN у майбутньому, і за останнє десятиліття комп'ютерний зір швидко розвивався завдяки збільшенню обчислювальної потужності, кращій обробці графіки та сучасним графічним процесорам. Це було прискорено завдяки створенню великого набору даних під назвою ImageNet, який містить 14 мільйонів зображень.

Завдяки цим досягненням моделі машинного навчання тепер набагато краще виділяють функції, ніж люди, і можуть ідентифікувати їх ефективніше,

оскільки вони можуть виявляти аномалії чи відмінності, які неможливо сприйняти традиційним людським програмуванням. І це має практичне застосування:

1) Сегментація зображення (Image Segmentation) - це поділ цифрового зображення на кілька сегментів зображення та позначення окремих пікселів, які складають кожен сегмент або об’єкт. Він широко використовується в автономних автомобілях. Приклад можна розглянути на рис. 3.2. Центральним для комп’ютерного зору є процес сегментації, який поділяє цілі зображення на групи пікселів, які потім можна позначити та класифікувати. Зокрема, семантична сегментація намагається семантично зрозуміти роль кожного пікселя в зображенні.

Рис. 3.2. Приклад сегментації зображень

2) Виявлення об’єктів (Object Detection) - це виявлення об’єктів на зображенні та його меж. Його часто використовують для виявлення дефектів, як тріщини на виробничій лінії або автоматизовують контроль якості, також це корисно у виявленні транспортних засобів на складах та в управління запасами складу. Традиційно наука про дані має величезну цінність для бізнесу. Це може допомогти компаніям у сфері маркетингу, у тому числі наприклад, націлювання на потрібних клієнтів. У медичній діагностиці комп’ютерний зір досягає такого стану, коли він може ідентифікувати певні

захворювання краще, ніж лікарі, оскільки вони більш помітні, який використовується для виявлення певних типів раку на медичних сканах. Машинний зір також використовується для боротьби зі зміною клімату.

Понад 200 мільйонів людей щороку знаходяться під загрозою повеней, ураганів, пожеж та інших стихійних лих, ймовірно, їх кількість збільшується через зміну клімату. Однак комп'ютерне бачення вже інтегровано із супутниковими знімками та камерами дронів, щоб відстежувати масштаби пошкоджень різних будівель і територій, спричинених стихійними лихами. Цю інформацію можна об'єднати з іншими даними та використати для визначення найкращих місць для встановлення тимчасових медичних шкіл і наметів, а також місць, де працівники гуманітарних організацій мають визначити пріоритетність коригувальних заходів.

Рис. 3.3. Приклад використання Object Detection в корисних сферах

Одне з найбільш захоплюючих застосувань комп'ютерного зору дозволяють власникам будинків фотографувати пошкодження своїх будинків після стихійних лих, дозволяючи їм краще прогнозувати витрати на ремонт і отримувати страхові виплати на місяці швидше, ніж до використання цих систем (рис. 3.3).

У найближчому майбутньому computer vision може сприяти підвищенню точності виявлення захворювань, а також підвищенню автоматизації трудомістких завдань, що призведе до кращого доступу пацієнтів до медичної допомоги.

3) Класифікація зображень (Image Classification) - класифікує різні групи об’єктів на зображенні шляхом присвоєння певних міток. Навчаючи модель комп’ютерного зору розпізнавати кілька класів об’єктів, модель може навчитися розпізнавати кожен із них і призначати нові невидимі зображення кожному класу. Наприклад, щоб дізнатись породу собаки по фото або ж визначити машину, яка порушила правила обмеження швидкості на дорозі загального користування [30].

Проблема класифікації зображень полягає в наступному: маючи набір зображень, кожне з яких позначено категорією, необхідно передбачити ці категорії для нового набору тестових зображень і виміряти точність передбачень. З цим завданням пов’язано декілька проблем, зокрема зміна точки зору, зміна масштабу, зміни в межах класу, спотворення зображення, оклюзія зображення, умови освітлення та фоновий шум.

Існує підхід в комп’ютерному зорі, керований даними, для вирішення цієї проблеми. Замість того, щоб намагатися визначити безпосередньо в коді, як виглядає кожна цікава категорія зображень, потрібно надавати комп’ютеру кілька прикладів кожного класу зображень, а потім розробляють алгоритми навчання, які вивчають ці приклади та вивчають візуальний вигляд кожного об’єкта.

Іншими словами, спочатку необхідно зібрати навчальний набір даних із анотованими зображеннями, а потім передати його на комп’ютер для обробки даних. Враховуючи цей факт, повну класифікацію зображення можна формалізувати так, що вхідні дані – це навчальний набір даних, що складається з N зображень, кожне з яких позначено одним із K різних класів, наступним кроком використовується цей навчальний датасет, щоб навчити класифікатор, як виглядає кожен клас. Кінцевим кроком, оцінюється якість класифікатора,

надати йому передбачити мітки для нового набору зображень, яких він ніколи раніше не бачив. На виході порівнюються реальні мітки цих зображень із тими, що передбачені класифікатором.

3.1.3. Зображення та його бачення для data engineering

Зображення представляє собою матрицю розміром M x N пікселів. В контексті інженерії даних і згорткових мереж також прийнято розглядати частину вхідного зображення, яку називають ядром і воно відоме як матриця згортки. Саме таке представлення добре підходить для завдань розмиття, підвищення різкості фото, визначення країв об'єктів та подібних завдань в обробці зображень.

На зображенні нижче (рис. 3.4) ми маємо матрицю зображення 5 x 5 у відтінках сірого, яка є жовтою, а матриця, яка є червоною 3 x 3, і є тим самим ядром, яке покращує загальне фото. Розмірність ядра має різні значення для різних завдань обробки зображень [31].

Рис. 3.4. Приклад позначення ядра зображення

Таблиця ядра згортається через великий масив, тобто масив зображень, зліва направо та зверху вниз, повертаючи одне значення пікселя на кожному

кроці згортки. Це значення одного пікселя є середнім значенням суміжних значень у сітці матриці 3 x 3. Нарешті, значення, що повертається на кожному кроці, є вихідною матрицею зображення. Наприклад, якщо необхідно оптимізувати певне зображення, можна використати визначений масив ядра для виконання цього завдання, що допоможе отримати на виході чітке фото. Таким чином виконується згортання для певної ділянки 3x3.

Для заданого зображення необхідно обрати координати (x, y) і вирівняйте центр таблиці ядра за цією координатою. Для матриці 3 x 3 центр буде 1 x 1. Потім перемножується кожне основне значення на відповідне значення матриці зображення та сумуються всі доданки. В ідеалі це сума множення на елементи масиву.

Різницю між лівим оригінальним зображенням та отриманим правим можна спостерігати після використання певного ядра (рис. 3.5). Для таких операцій, як розмиття, визначення країв та інших завдань із зображенням, ядро має різні значення [32].

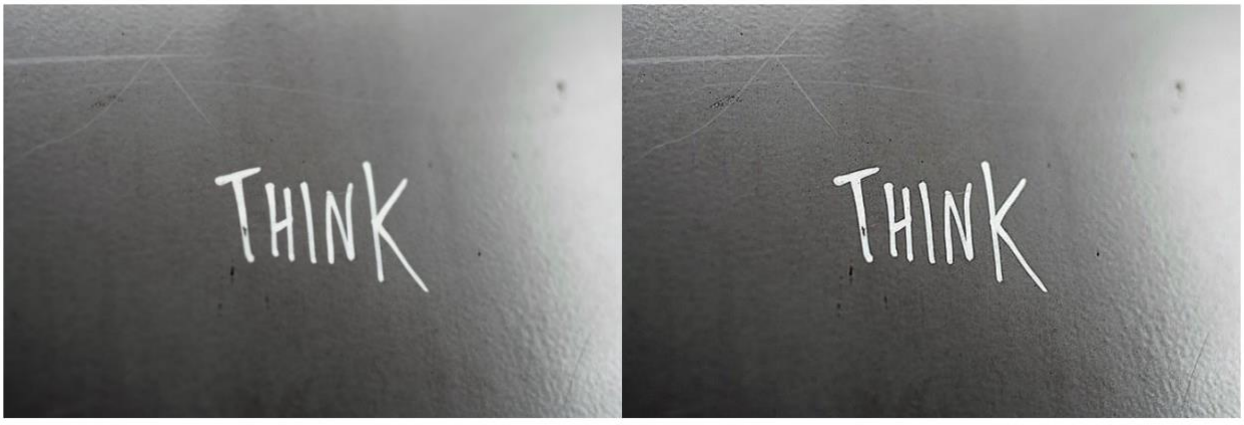


Рис. 3.5. Приклад застосування ядра для покращення різкості у фото

3.1.4. Згорткові нейронні мережі та розуміння їх роботи в data engineering

Маючи певну базу як можна представляти зображення в науці про дані, що описано вище, можна розглянути роботу CNN з середини і зрозуміти, чому вони такі ефективні в роботі з фото/відео інформацією.

Головним плюсом у використанні згорткових нейронни мереж в deep learning є способом зменшити кількість параметрів при навчанні. Замість того, щоб мати справу з повністю підключеною мережею, підхід CNN використовує доступні параметри повторно. Основна ідея CNN полягає в тому, що необхідно мати лише місцеве розуміння зображення, тобто ядро. Практична перевага полягає в тому, що менша кількість параметрів значно скорочує час навчання та зменшує кількість даних, необхідних для навчання моделі.

Замість повноцінної мережі, яка зважує кожен піксель, CNN має достатню вагу, щоб переглянути невелику частину зображення. Це як читати книгу через збільшувальне скло, зрештою ви читаєте всю сторінку, але в той же час ви дивитеся лише на невелику частину сторінки.

Прикладом може слугувати зображення розміром 256 × 256. Замість того, щоб обробляти все зображення відразу, згорткова нейронна мережа може ефективно сканувати його частину за частиною, тобто використовувати ядро розміром 5x5. Сектор 5 × 5 ковзає по зображенню, зазвичай зліва направо та зверху вниз, як показано на зображенні нижче (рис. 3.6). Швидкість, з якою мережа «ковзаєте», називається довжиною кроку. Наприклад, розмір кроку 2 означає, що ковзне ядро 5x5 переміщується на 2 пікселі за раз, доки не охопить усе зображення. Цей сектор 5x5 пов’язаний з ваговою матрицею 5x5.

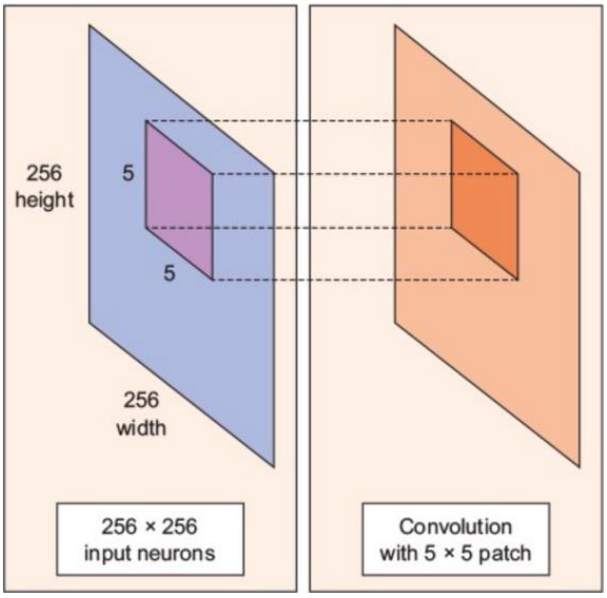


Рис. 3.6. Приклад роботи CNN з використанням ядра

Механіка ядра, що «ковзає», відбувається на згортковому шарі нейронної мережі. Типовий CNN має багато шарів переплетення. Кожен з яких зазвичай створює багато витків, що чергуються, тому вагова матриця є тензором 5 × 5 × n, де n – кількість витків.

Наприклад, припустимо, що зображення проходить через площину згортки у ваговій матриці 5 × 5 × 64. Воно створює 64 витки, пересуваючи ядро 5 × 5. Отже, цей шаблон має параметри 5 × 5 × 64 = 1600 , що набагато менше параметрів, ніж у повністю підключеної мережі 256 × 256 = 65 536.

Також принадність CNN полягає в тому, що кількість параметрів не залежить від розміру вхідного зображення. Ви можете запустити ту ж згорткову нейронну мережу на зображенні 300 x 300 пікселів та кількість параметрів у шарі згортки не зміниться.

3.1.5. Підготовка даних.

Тренування та розвертання моделі згорткової нейронної мережі нині можливе за лічені хвилини. По факту кожен може просто завантажити свої дані та натиснути кілька кнопок, щоб навчити та реалізувати свою модель через API популярних фреймворків для навчання моделей [33]. На сьогоднішній день, дані є найскладнішим пунктом в реалізації необхідної CNN для виявлення об’єктів, оскільки:

- для виявлення об’єктів потрібні набагато більші набори даних, ніж для традиційних класифікаторів зображень;
- для формування датасетів потрібні спеціалісти в сфері інженерії даних, щоб надати для тренування моделі достатньо повноцінний по ємкості набір даних, що може надати CNN всю необхідну інформацію для якісної оцінки;
- будь-які елементи, які ШІ потрібно розпізнавати, спочатку мають бути належним чином позначені тегами відповідних класів, тобто вони певним чином позначені та обмежені;

- розмітка даних – це тривалий і виснажливий процес, який часто потребує команди людей;
- необхідно бути абсолютно послідовними при розмітці даних, а люди, що цим займаються, мають бути добре навченими та скоординованими.

Вибір типу архітектури просто збільшує або зменшує загальну точність лише на кілька відсотків. Шкода, що більшість досліджень машинного навчання зосереджено на архітектурному дизайні, нових тригерних функціях і шарах. Крім того, дослідження зазвичай проводяться знову і знову на тих самих величезних наборах даних, наприклад як ImageNET, COCO, Google Open Images. В основному дослідницькі групи та великі технологічні компанії витрачають мільйони доларів на цю частину дослідження, але не на процес підтримки даних.

Зосередження на даних дає набагато кращі результати, ніж останні архітектури, розроблені Google. Більшість проектів машинного навчання починаються з невеликої кількості даних або без них, що часто приводить до поганого функціонування системи або ж взагалі виходять з ладу і не можуть надалі виконувати поставлені перед собою задачі.

Дослідник штучного інтелекту Ендрю НГ закликає спільноту машинного навчання більше зосереджуватися на даних, і з цим не можна не погодитися. Набагато кращим способом є поступове уточнення даних і збалансування датасету. [34] Це допомагає заощадити багато часу на навчанні, зосередившись більше на даних, ніж на моделі. Тестування різних архітектур також вимагає тонкого налаштування багатьох гіперпараметрів.

При розмітці зображень, людям потрібно переглядати результати роботи інших, щоб це дозволило підтвердити точність наданої розмітки.

Отже, беручи курс на підготовку великої кількості даних, нам потрібно знати, які дані вже перевірено, а це може складати сотні тисяч зображень, тисячі категорій, включаючи виявлення об'єктів і сегментацію.

Отримуючи все нові та нові дані команда з розмітки зосереджуються на головному типі проблеми, яку необхідно вирішити. Проте варто завжди

пам’ятати, що вирішення однієї проблеми створює іншу або знижує точність у різних частинах системи. Цей фактор дає впевненість в тому, що потрібно бути дуже обережними щодо правильного балансу правильних шаблонів зображень. Тому потреба систематично вдосконалюємо набір навчальних даних є постійною.

Інструменти анотації підтримує перевірку даних і кілька інструментів малювання, таких як прямокутник, багатокутник і класичне маркування, що допомагає команді з розмітки підготувати максимально відповідні обмежувальні рамки для кожного об’єкта. Процес анотації з часом пришвидшується. Це відбувається тому, що можна просто навчити модель виявлення об’єктів знаходити необхідні класи, а потім розмітка зводиться до поправлення наявних bboxs або додавання нових, що нейронка не змогла знайти. За допомогою MLOps системи анотацій є можливість провести A/B тестувати різних версії моделей, навчених на різних версіях набору даних. В результаті, маючи декілька вихідних даних необхідно порівняти їх та визначити найкращий, щоб продовжити лейблінг максимально ефективно.

У роботі кожної моделі присутні параметри оцінкі якості роботи моделі, основними параметрами є precision та recall, що в свою чергу означає точність та відгук. У деяких проектах доводиться розпізнавати різні типи об’єктів, які виглядали дуже схожими. Модель дуже складна для вивчення. Тому що, якщо у присутні різні класи, які виглядають дуже схожими, функція втрати виявлення об’єктів зрештою застрягне в ранніх епохах. Наприклад, визначити сотні видів птахів може бути дуже важко.

Проблему виявлення багатьох типів об'єктів можна розділити на два етапи. Спочатку необхідно знайти об’єкт, а потім класифікувати вміст обмежувальної рамки. Варто зважати, що чим більше у класів розпізнавання підготовлено для нейронки, то більше даних потрібно зібрати. Розбиття проблеми на меншу кількість класів є дуже ефективним способом. Крім того, це не зіпсує жодну іншу частину системи, оскільки кожна нейронна мережа бачить лише частину даних, яка їй передається. Якщо робота проводиться з

великою моделлю, усі зображення повинні мати всі межі та мітки, а також вони повинні бути добре збалансованими, щоб один клас не був домінуючим на іншими, щоб це не призводило дисбалансу в роботі мережі.

Ще одним важливим фактором у роботі з великими датасетами є місце їх зберігання. Таким місцем є сховище даних – це простір для зберігання великих обсягів інформації, як поточних, так і минулих, для аналізу даних. Ці дані надходять з різних джерел.

Для роботи з простором для зберігання інформації та джерелом інформації необхідно розібратись з інструментами ETL – витягування, доставка та загрузка даних. Тому ETL — це процес, за допомогою якого дані витягуються з джерела, а потім перетворюються в читабельний формат для аналізу даних і зберігання в сховищах. Аналіз даних, які відповідають певному бізнес-запиту, використовує «пакетну обробку», щоб допомогти кінцевому користувачеві, а ETL витягує дані з різних джерел, потім застосовує певні правила відповідно до потреб бізнесу, і, нарешті, перетворені дані завантажуються в базу даних або платформу бізнес-аналітики, щоб вони були доступні в будь-який час будь-якій організації.

3.1.6. Проблеми в data engineering.

Хоча інженерія даних є новою сферою, інженери даних все ще стикаються з деякими проблемами:

1) Проблеми з якістю даних. Зі збільшенням обсягів даних стає складніше керувати великими обсягами даних. Тут виникає проблема якості даних, оскільки кількість даних, отриманих із багатьох джерел сьогодні, не є чистою та містить багато невідповідностей. Тому завдання інженера даних полягає в тому, щоб очистити відповідні дані та надати їх спеціалісту з обробки даних.

2) Тестування каналу даних – навіть найменший витік може спричинити великі втрати бізнесу.

3) Перемикання контексту: інколи запуск завдання ETL стає дуже трудомістким процесом, і під час виконання обов'язково виникають помилки. Тому важко повернутися до мислення та перейти до наступної ітерації.

4) Узгодження. Інженери з даних є основою будь-якого наукового проекту з обробки даних, тому через суперечливі дані можуть виникати проблеми. У великій організації необхідно створити послідовність і узгодженість, щоб максимально вирішити це питання [35].

3.1.7. Масштаби та майбутнє технології обробки даних

Зі зростанням технологічного прогресу інженерія даних знаходиться на порозі повної трансформації, оскільки поточні розробки включають штучний інтелект, машинне навчання, безсерверні обчислення та гібридні хмари, тощо. Використання великих даних та різноманітних інструментів науки про дані збільшиться в найближчі роки. Коли ми переходимо від пакетно-орієнтованої переміщення та обробки даних до переміщення та обробки даних у реальному часі, ми бачимо трансформацію до «потоків даних та систем обробки даних у реальному часі». Сховища даних нещодавно стали дуже популярними, оскільки вони достатньо гнучкі, щоб за потреби розміщувати центри обробки інформації, «озера» або окремі набори даних.

Наступні чотири області були визначені як технологічні досягнення в інженерії даних майбутнього [36]:

1) Пакетне перетворення у режимі реального часу: зміни систем збору даних швидко замінюють пакетний ETL і перетворюють потокову передачу даних у реальність. Традиційні функції ETL тепер виконуються в реальному часі.

2) Розширений зв'язок між джерелами даних і сховищами даних.

3) Аналітика самообслуговування за допомогою інтелектуальних інструментів на базі технології обробки даних.

4) Гібридні архітектури даних, що охоплюють локальні та хмарні середовища.

3.2. Використовувані технології та фреймворки для навчання згорткових нейронних мереж.

3.2.1. Python.

Першим і головним інструментом у машинному навчані – мова програмування, яка має бути гнучкою та давати всі необхідні можливості для покращення виконання доцільних задач. Python пропонує це все, саме тому на сьогоднішній день так багато проектів в ШІ створенні саме з його використання.

Від розробки до розгортання та обслуговування, Python допомагає розробникам бути продуктивними та впевненими щодо програмного забезпечення, яке вони створюють. Переваги, завдяки яким мова програмування найкраще підходить для машинного навчання та проектів на основі ШІ, включають простоту та узгодженість, доступ до чудових бібліотек і фреймворків для ШІ та машинного навчання, гнучкість, незалежність від платформи та широке співтовариство. Це додає загальної популярності мови.

Python пропонує стислий і читабельний код. Хоча за машинним навчанням і штучним інтелектом стоять складні алгоритми та різноманітні робочі процеси, простота цієї мови дозволяє розробникам створювати надійні системи. Замість того, щоб зосереджуватися на технічних нюансах мови, розробники докладають усіх зусиль для вирішення проблеми ML.

Крім того, Python привабливий для багатьох розробників, оскільки його легко вивчити та код написаний на цій мові зрозумілий людям, що полегшує створення моделей для машинного навчання.

Багато програмістів кажуть, що Python більш інтуїтивно зрозумілий, ніж інші мови програмування. Інші вказують на безліч фреймворків, бібліотек і розширень, які спрощують реалізацію різних функцій. Загальновизнано, що Python підходить для спільного впровадження, коли задіяно кілька розробників. Оскільки Python є мовою загального призначення, він може виконувати низку складних завдань машинного навчання та дає змогу швидко

створювати прототипи, які дозволяють тестувати ваш продукт для цілей машинного навчання.

Впровадження алгоритмів AI і ML може бути складним і вимагає багато часу. Дуже важливо мати добре структуроване та перевірене середовище, щоб дозволити розробникам пропонувати найкращі рішення для програмування.

Щоб скоротити час розробки, програмісти звертаються до ряду фреймворків і бібліотек Python. Мова зі своїм багатим набором технологій має великий набір бібліотек для штучного інтелекту та машинного навчання . Ось деякі з них:

- Keras, TensorFlow і Scikit-learn для машинного навчання;
- NumPy для високопродуктивних наукових обчислень і аналізу даних;
- SciPy для передових обчислень;
- Pandas для аналізу даних загального призначення;
- Seaborn для візуалізації даних;

Scikit-learn містить різноманітні алгоритми класифікації, регресії та кластеризації, включаючи машини опорних векторів, випадкові ліси, посилення градієнта, k-середні та DBSCAN, і розроблено для роботи з числовими та науковими бібліотеками NumPy та SciPy.

За допомогою цих рішень можливо швидше розробляти продукт. Команді розробників не доведеться заново винаходити колесо, і вона може використати наявну бібліотеку для впровадження необхідних функцій (табл. 3.1).

Таблиця 3.1

Найпоширеніші рішення для ШІ в Python

Назва сфери використання:	Рішення:
Аналіз та візуалізація даних	NumPy, SciPy, Pandas, Seaborn
Машинне навчання	TensorFlow, PyTorch, Scikit-learn

Продовження таблиці 3.1

Назва сфери використання:	Рішення:
Комп’ютерний зір	OpenCV
Обробка природної мови	NLTK, spaCy

3.2.2. FiftyOne

FiftyOne — це набір інструментів машинного навчання з відкритим вихідним кодом, який дає змогу командам із вивчення даних покращувати продуктивність своїх моделей комп’ютерного зору, допомагаючи їм керувати високоякісними наборами даних, оцінювати моделі, знаходити помилки, візуалізувати вбудовування та швидше переходити до виробництва (рис. 3.7).

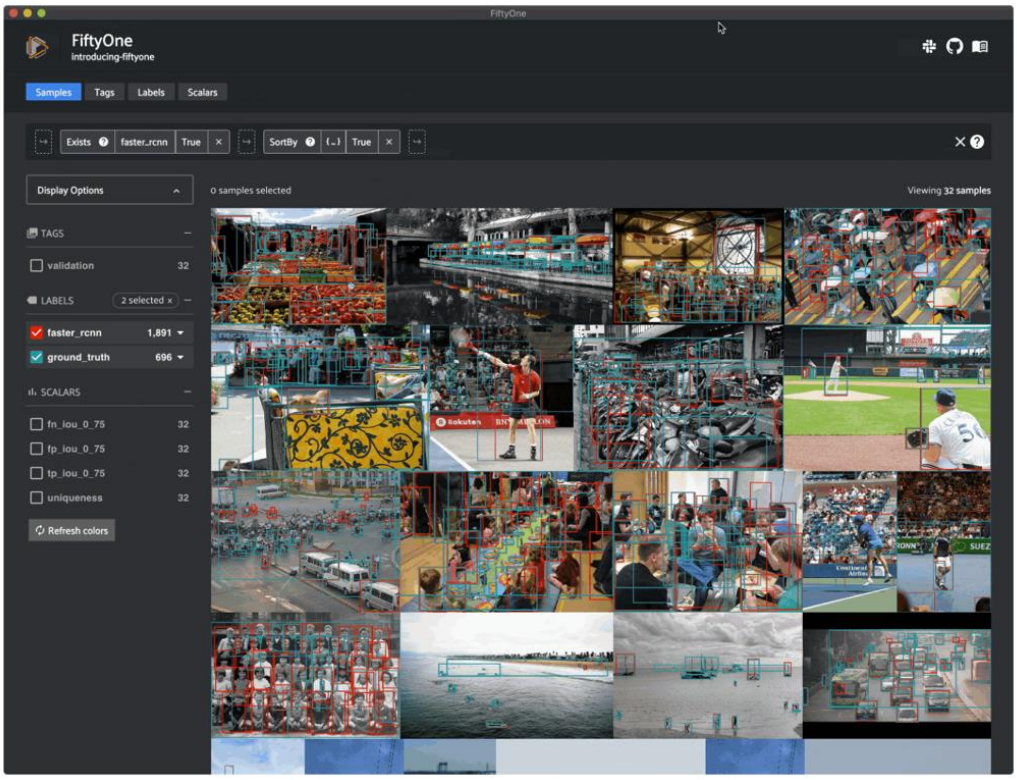


Рис. 3.7. Приклад інтерфейсу фреймворка FiftyOne

FiftyOne усуває зусилля, необхідні для швидкого завантаження та візуалізації даних у різноманітних поширених або спеціальних форматах. Хоча це завдання, безумовно, здійсненне будь-яким інженером ML, потрібні серйозні зусилля, щоб належним чином об’єднати різні формати

даних/анотацій у стандартний формат, з яким вони можуть працювати. FiftyOne усуває ці зусилля та зустрічає інженерів там, де вони вже працюють, на Python.

FiftyOne не просто візуалізує дані, він також надає потужні утиліти аналізу набору даних [37]. Візуалізація наборів даних у додатку та потужність бібліотеки Python самі по собі вже корисні для інженерів CV та ML, також в FiftyOne розроблені функції для забезпечення звичайних робочих процесів, реалізація яких є важливою, але складною сьогодні:

- перетворення між форматами набору даних;
- курування різноманітних і репрезентативних наборів даних;
- вибір відеокадрів для анотації та навчання моделей зображень;
- автоматичний пошук помилок міток;
- оцінка прогнозів моделі;
- виявлення та візуалізація режимів відмови у ваших моделях.

3.2.3. Аналіз інструментів для розмітки

Робота над тренуванням моделі завжди розпочинається з підготовки великого набору даних, тому важливо розглянути інструмент для відпрацювання на наступному кроці – застосунок для розмітки зображень. На ринку представлено безліч додатків, які закривають певні задачі і мають свої особливості по відношенню до конкурентів. Варто розглянути найбільш популярні застосунки, бо зазвичай у топових рішень є всі необхідні рішення та зручні додаткові функції. Порівнюватись буде чотири системи, а саме: Supervise.ly, Hasty.ai, CVAT.ai, Make-Sense.ai (табл. 3.2).

Таблиця 3.2

Порівняння додатків для розмітки даних

Параметр порівняння	Supervise.ly	Hasty.ai	CVAT.ai	Make-Sense.ai
Зручність та ефективність роботи	+	+	+	+

Продовження таблиці 3.2

Параметр порівняння	Supervise.ly	Hasty.ai	CVAT.ai	Make-Sense.ai
Веб-додаток	+	+	+	+
Локальне розвертання	-	-	+	+
Підтримування різних форматів датасетів	+	+	+	+
Автолейблінг CNN	-	-	+	+
Різні типи розмітки	+	+	+	+
Різновид форм обмежувальної рамки	+	+	+	+
Експорт розмітки у COCO форматі	+	-	+	-
Підтримка роботи з відео	-	-	+	-
Інтеграція з FiftyOne	-	-	+	-

Розглядаючи ці чотири програмні рішення, можна побачити їх масштабність та великий набір функцій, проте унікальності, які необхідні в роботі виділяють саме застосунок для маркування зображень – CVAT [38]. Кожен з порівнюваних додатків зручний та придатний для роботи, але вирішальним фактором є експорт розмітки у форматі COCO та інтеграція з фреймворком FiftyOne.

3.2.4. OpenCV

Кожне застосування комп'ютерного бачення для вирішення певних задач не залишає в стороні потужний інструмент для опрацювання медіа даних та потокової інформації – OpenCV. Це рішення, яке реалізоване на багатьох популярних мовах, надає великий функціонал для опрацювання фото та відео контенту і є одним з перших помічників інженера даних з обробки та підготовки датасету для навчання моделі.

OpenCV випущено за ліцензією BSD, що робить його безкоштовним для академічного та комерційного використання. Він має інтерфейси C++, C,

Python і Java і сумісний з Windows, Linux, Mac OS, iOS і Android. OpenCV розроблено для підвищення продуктивності обчислень і зосереджено на програмах реального часу. Бібліотека, написана на оптимізованому C/C++, може використовувати багатоядерну обробку. Увімкнувши OpenCL, ви можете скористатися перевагами апаратного прискорення платформи [39].

Зображення можна представити у вигляді багатовимірного масиву. Це тому, що машина може представляти будь-що у вигляді чисел, і в Python ви можете використовувати NumPy для представлення цього.

```
1  import cv2
2  image = cv2.imread("00-puppy.jpg")
3  img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
4  plt.imshow(img_rgb)
```

image_read.py hosted with ❤ by GitHub view raw

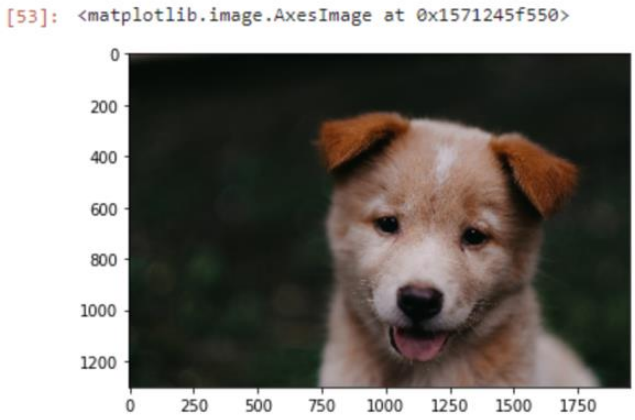


Рис. 3.8. Приклад використання OpenCV

Для зображень часто використовується загальне слово під назвою пікселі або значення пікселів. Є три кольорові канали для кольорових зображень. Таким чином, кольорові зображення мають кілька значень на піксель. Ці матриці можуть відрізнятися за розміром залежно від роздільної здатності та глибини кольору. Значення кольорів варіюються від 0 до 255. Ці кольорові канали зазвичай представлені як червоний, зелений, синій – RGB. Для прикладу, можна показати легкість роботи з зображеннями, де видно як за декілька рядків коду можна прочитати зображення та відобразити його у

правильному форматі (рис. 3.8). Це показує головний принцип використання OpenCV в машинному навчанні – легкість використання. Саме це допомагає розробникам приділити більше часу вирішенню поставленій задачі, а ніж опрацювання даних.

3.2.5. Порівняння фреймворків для навчання згорткових нейронних мереж.

На сьогодні є два найпопулярніші рішення для тренування нейронних мереж, в тому ж числі і CNN – це TensorFlow та PyTorch. Їх і варто розглядати при виборі фреймворку для навчання, вони вже мають достатньо можливостей для вирішення більшості поставлених задач, проте розробники доповнюють кожного з них все більше і більше новими функціями [40].

TensorFlow розроблений компанією Google Brain і активно використовується в дослідницьких і виробничих цілях самими розробниками. Його попередник із закритим кодом називається DistBelief.

PyTorch є двоюрідним братом фреймворку Torch на основі Lua, розробленого та використовуваного Facebook. Однак PyTorch — це не просто набір обгорток для підтримки популярної мови, його було переписано й оптимізовано, щоб бути швидким і виглядати максимально зрозумілим для розробників на мові програмування Python.

Порівняння цих інструментів зводиться до пошуку найоптимальнішого рішення під конкретну задачу, яка стоїть перед розробниками.

Таблиця 3.3

Порівняння TensorFlow та PyTorch

Параметр оцінки:	TensorFlow	PyTorch
документація	добре задокументований, перевірений часом	документація достатня, проте збільшується з кожною версією

Продовження таблиці 3.3

Параметр оцінки:	TensorFlow	PyTorch
налагодження	максимально консервативно через tfdbg	гнучко та динамічно, немає обмежень у інструментах
відображення	інтегрований інструмент Tensorboard, який є повноцінним та добре спроектованим	не повний, проте зручний у використанні visdom, є можливість інтеграції з іншими рішеннями
розгортання моделі	зручний та незамінний інструмент TenserFlow Serving для розмертання моделі на gRPC сервері	готового рішення немає, проте можна використовувати Flask або інші інструменти для веб-додатків
паралелізм даних	Механізм паралелізму необхідно налаштовувати та адаптувати власноруч	присутній декларативний паралелізм з простою можливістю використання
складність реалізації	низькорівнева абстракція, потрібно багато реалізовувати самостійно	високорівнева абстракція, за допомогою якої можна швидше отримати результат

З порівняння вище (табл. 3.3) можна зробити висновок, що кожен з цих фремворків має все необхідне для роботи та багато додаткових можливостей для спрощення або ускладення навчання в залежності від задачі, тому вибір бібліотеки зазвичай виконується за бажанням самого розробника.

Розвиток штучного ітелекту та комп’ютерного зору набрали свої оберти та розвиваються дуже швидкими темпами, саме тому великої різниці, який з фреймворків обирати, немає. Кожен з цих інструментів допомагає створювати масштабовані додатки та якісні нейронні мережі.

ВИСНОВКИ ДО РОЗДІЛУ 3

Інженерія даних передбачає створення, проектування та налаштування конвеєрів даних і перетворення даних для наукових спеціалістів/розробників великих даних, щоб зробити їх більш зручними для користувачів. Розробка даних, зосереджена насамперед на практичних наслідках науки про дані, є на крок попереду збору та аналізу основних технічних даних. Інженери даних відповідають за проектування архітектури даних, розробку процесів набору даних, моделювання даних та видобуток даних.

Інженери даних тісно співпрацюють з аналітиками даних і бізнес аналітиками, щоб трансформувати дані перед їх обробкою та розробляти моделі для роботи. Сховища даних і «озера» даних зазвичай використовуються для зберігання великих даних. «Озеро» даних — це величезне сховище необроблених і неструктурованих даних, тоді як сховища даних об’єднують структуровані, відфільтровані та оброблені дані.

Проблеми з якістю даних, витоки даних і перемикання контексту є одними з найбільших проблем у data engineering. Тим не менш, інженерія даних вважається дуже затребуваною сферою в усьому світі. Майбутнє розробки даних дуже динамічне, оскільки воно пропонує аналіз і обробку даних у реальному часі. Big data та інші інструменти data science будуть широко використовуватися в майбутньому.

Підходячи до питання розробки та налаштування пайплайну підготовки датасету, завжди необхідно виходити з позиційної задачі моделі, що вона має виконувати. Інженерія даних перетинається з багатьма супутніми сферами.

РОЗДІЛ 4. НАВЧАННЯ ТА ТЕСТУВАННЯ НЕЙРОННОЇ МЕРЕЖІ

4.1. Постановка задачі

Тренування нейронної мережі складається з багатьох частин, послідовність яких і складає пайплайн навчання. Найпершим кроком в цьому є описання проблематики завдання та повного формулювання цілі, які буде вирішувати отримана на виході нейронна мережа. Саме ціль робочої системи буде вирішувати, які методи та дані варто використовувати. В даній роботі висвітлюється робота над мережею, що буде покращувати та полегшувати роботу у підготовці потокової інформації для розробника даних. А саме покращувати якість використовуваних даних, що будуть у майбутньому використовуватись в комп'ютерному зорі.

Цілі нейронної мережі:

1. Аналіз отриманих даних у вигляді фото.
2. Знаходження необхідних об'єктів на кожному екземплярі потокових даних.
3. Видання точних координат класу «front_sign».

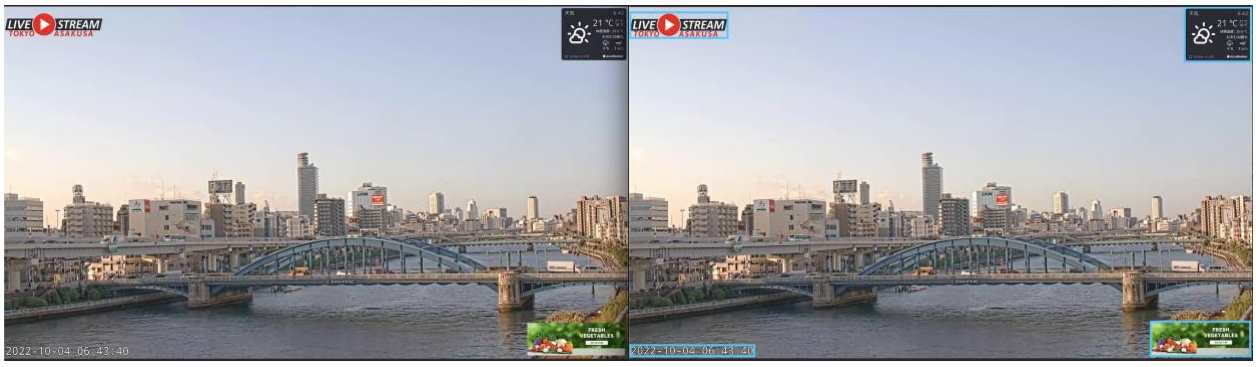


Рис. 4.1. Приклад виконання цілей роботи нейронки.

Описуючи роботу майбутньої нейронки, можна виділити такий функціонал – знаходження на фото всіх написів та рекламних банерів, які заважають повністю сприймати відображувану інформацію (рис. 4.1). Саме тому в навчальних даних для тренування нейронки в комп'ютерному зорі необхідно правильно підготовлювати інформацію. Повноворматний розмір зображення на рис. 4.1 можна побачити в додатку А.

4.2. Підготовка даних для тренування

Другим не менш важливим кроком є аналіз проблематики знаходження даних для побудування датасету. Кожна система повинна отримати достатню кількість «якісної» та валідної інформації для створення певних закономірностей і паттернів для роботи у необхідній для роботи ситуації. Не виключенням є і розробка даної системи.

Пошук даних має два напрямки, кожен з яких дає унікальні набори інформації, що допоможуть зробити нейронку більш універсальною та «сильною» на виході. Орієнтуючись на роботу та в яких умовах має працювати розроблений додаток, потрібно зважати і на відповідність даних, саме тому було вирішено збирати датасет з відкритих джерел, а саме відео стрімінговий хостинг youtube.com та пошукову систему google.com. Саме ці два «сховища» зберігають в собі зображення максимально схожі до реального «життя», що дає змогу створити максимально релевантну нейронну мережу для відпрацювання на реальних даних.

Для збору даних з сервісів було розроблено декілька термінальних додатків, які використовували мережеве з'єднання для отримання інформації з їхніх датацентрів.

Збір даних з YouTube відбувався в два етапи, потрібно було знайти максимально релевантні відео, які б мали в собі відповідні до нашої цілі. Зібравши достатню кількість посилань не відео (рис. 4.2), а саме більше чотирьох тисяч, було написано скрипт, який отримує відео потік за кожним посиланням зі списку і збирає по одному фрейму зі всього отриманого потоку.

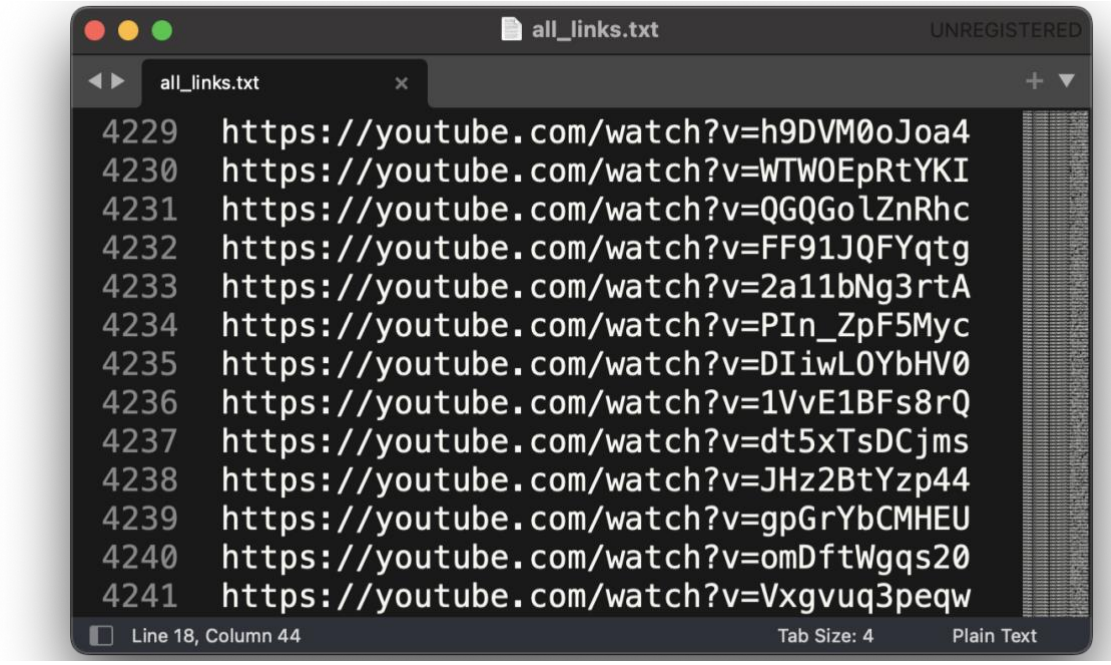


Рис. 4.2. Зібрані посилання з ютубу

Для кращого розуміння необхідних даних для тренування, потрібно розділити два поняття:

1) Задній фон (background, bg) – це основна інформація, яку в собі містить потокове відео;
2) Передній фон (foreground, fg) – це інформація, яка була накладена на основний потік заднього фону, в даному випадку необхідні написи на фреймі та рекламні банери.

Маючи розуміння даних, на виході отримуємо три тисячі унікальних фото, які ми можемо використати у навчанні моделі. Написаний програмний додаток представлено в Додатку Б.

Наступним кроком був збір даних з пошукового сервісу google.com. Серед мільйонів фото, які містить в собі ця пошукова система, було зібрано біля 5 тис фото, щоб мати можливість відфільтрувати необхідні для навчання дані та невалідні зображення. Збір фото собою представляє збереження посилання на пошуковий запит для подальшого парсингу веб-сторінки та отримання прямого посилання на зображення за допомогою написаного скрипта для скрапінгу, який представлений в додатку В.

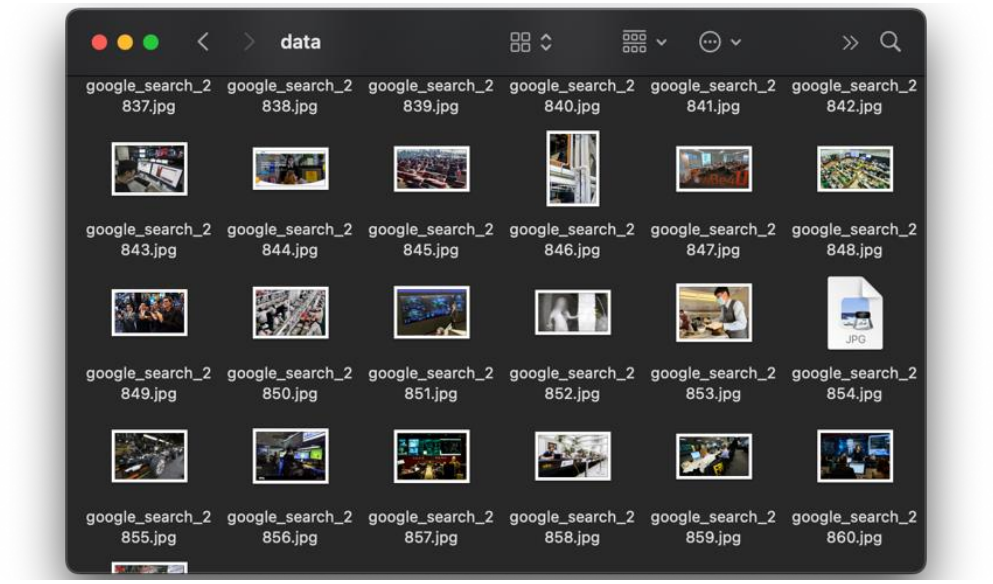


Рис. 4.3. Отримані зображення з google.com

Суть в тому, що скрипт симулює роботу браузера за допомогою бібліотеки Selenium і переходить за отриманим URL, далі читаючи HTML-розмітку, отримуються якорі-вказівники на оригінальні фото. Маючи набір унікальних посилань за допомогою бібліотеки requests було отримано байтове представлення кожного зображення та збережено у jpg-форматі для подальшого використання.

4.3. Формування датасету

Маючи в загальній кількості 9 тисяч фото, потрібно скомпанувати їх в датасет. Це можливо зробити за допомогою бібліотеки Fiftyone. Для почетку необхідно створити об’єкт класу Dataset та налаштувати його під необхідні задачі (рис. 4.4). Після створення датасету, було додано назву класу за замовчуванням, який необхідно мати, а саме клас – front_sign.

```
dataset = fo.Dataset()
dataset.default_classes = CLASS_LIST
dataset.save()
```

Рис. 4.4. Створення датасету за допомогою FiftyOne

Тепер маючи віртуальне сховище, що буде описувати та об’єднувати всі зображення, необхідно отримати кожен екземпляр даних та створити sample – класс, що описує зображення та всю відповідну для нього інформацію (рис. 4.5).

```
samples = []
for path in glob.glob(INPUT_IMAGES_FOLDER + "/*.jpg"):
    sample = fo.Sample(filepath=path, ground_truth=fo.Detections(detections=[]))
    sample["ground_truth"] = fo.Detections(detections=[])

    samples.append(sample)

dataset.add_samples(samples=samples)
dataset.save()
```

Рис. 4.5. Додавання до датасету кожен екземляр зображення

Тепер маючи готовий датасет необхідно його зберегти та експортувати у необхідному форматі (рис. 4.6). Працюючи зі згортковими нейронними мережами yolov5, варто оперувати максимально зручним для роботи з ними форматом, а саме формат COCO.

```
print("Saving dataset!")
dataset.export(
    dataset_type=fo.types.COCODetectionDataset,
    labels_path=INPUT_LABELS_FOLDER,
    label_field="ground_truth")
```

Рис. 4.6. Експорт створеного датасету

При експорті датасету утворюється файл у json-форматі, який містить інформацію про кожне зображення і, надалі, маючи цей файл, буде можливість отримати всю необхідну та зроблену для фото розмітку.

При приготуванні набору даних, необхідно підтримувати структуру збереження інформації для подальшого зручного використання, а також для правильного співвідношення метадати json-файлу і кожного зображення (рис. 4.7).

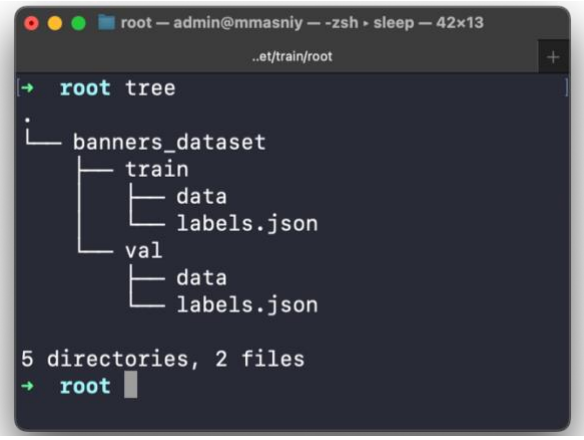


Рис. 4.7. Приклад правильної структури організації файлів у датасеті

4.4. Розмітка навчальних даних

Наступним важливим кроком у створенні навчальних даних, має бути розмітка підготовлених зображень у відповідності до цілі роботи згорткової нейронної мережі, а саме у відповідності до необхідних класів об’єктів. В даному випадку буде використовуватись вказаний за замовчуванням клас - front_sign.

Для вигрузки датасету на сервіс для розмітки, використаємо інтеграцію FiftyOne та CVAT, що допоможе контролювати завантажені набори даних до розмітки та вигрузити їх при необхідності.

Перед використанням софту для розмітки, варто розділити отриманий набір даних на тренувальну частину та валідаційну (рис. 4.7). При навчанні моделі варто розуміти принцип оцінювання «якісної» роботи нейронної мережі, а це відбувається за допомогою «проганяння» тестової виборки через отримані ваги на виході. Таким чином можна сказати, що результат роботи також залежить від правильного валідування CNN, тому що маючи максимально різноманітний і наближений до реального використання моделі тестовий набір даних можна наблизити до «ідеального» стану робочу систему.

Ще одним важливим фактором є унікальність частин train та val, тобто в другому наборі унікальні дані по відношенню до першого і немає перетину. Це не дає можливості нейронці на ранніх етапах при навчанні сформувати признаки, за якими буде відбуватись пошук об’єктів при тестуванні.

На даному етапі, маючи навчальну та тестову вибірку, необхідно перейти до етапу розмітки даних. Перед цим використовується скрипт, який написаний для інтеграції з CVAT, для передання даних до наступного кроку пайплайну підготовки датасету. Повний лістинг програми буде знаходитись у додатку Г.

```
def CVAT_upload():
    dataset = import_base_dataset()
    dataset_len = len(dataset)
    dataset_parts = int(dataset_len / SUB_LEN)

    if dataset_len % SUB_LEN != 0:
        dataset_parts += 1

    print("Dataset len:", dataset_len, "parts:", dataset_parts)

    for i in range(dataset_parts):
        if (i * SUB_LEN) + SUB_LEN > dataset_len:
            sub_max = dataset_len
        else:
            sub_max = (i * SUB_LEN) + SUB_LEN

        dataset_new = dataset[i * SUB_LEN:sub_max].select_fields("ground_truth").clone()

        dataset_new.name = NAME + SUBNAME + '_part_' + str(i)

        dataset_new.persistent = True

        dataset_new.annotate(anno_key + '_part_' + str(i), label_field="ground_truth", url="https://app.cvat.ai/mem",
                             project_name=NAME)
        print("Uploading dataset:", NAME + SUBNAME + '_part_' + str(i))

        time.sleep(2)

    print("list of datasets:", fo.list_datasets())
```

Рис. 4.8. Вигрузка датасета в CVAT

На рис. 4.8 представлено функцію, за допомогою якої відбувається вигрузка набору даних. Принцип роботи програми заключається в тому, що необхідно імпортувати створений раніше датасет, потім розбиваємо набір даних на пакети по 300 фото для зручності роботи. А вигрузка відбувається за допомогою підмодуля всередині FiftyOne, мотод називається annotate, в який передається назва пакету, вид розмітки, в даному випадку ground_truth, та адресу сервісу по розмітці.

Після вигрузки датасетів, в CVAT створюється проект, який узагальнює всі вигружені пакети в одну загальну частину (рис. 4.9).

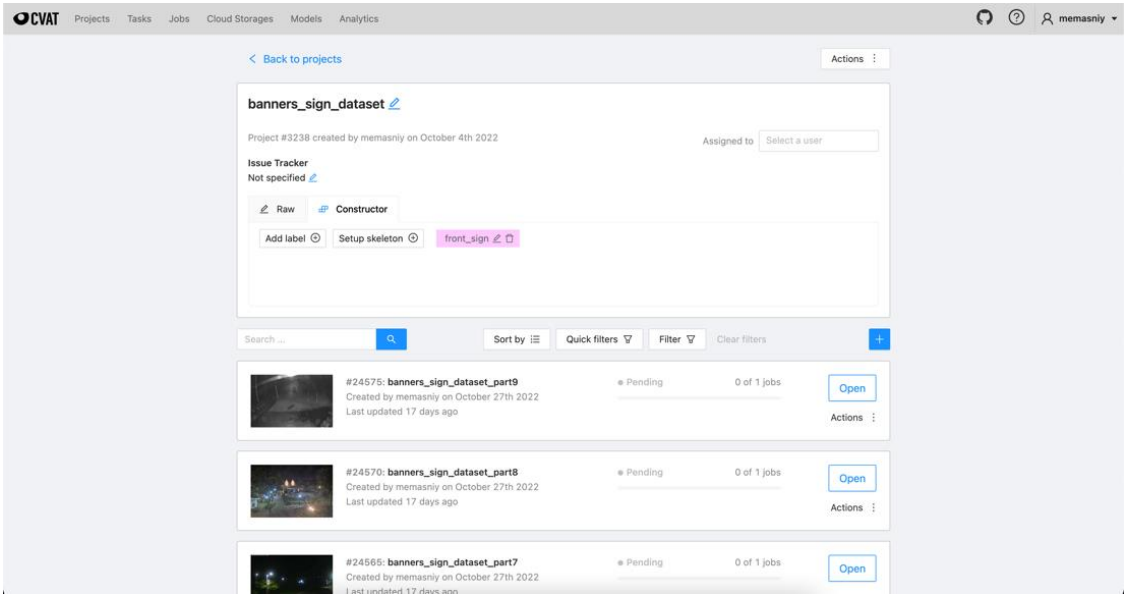


Рис. 4.9. Створений проект в CVAT

В самому проекті можна редагувати частини, при необхідності можна додати пакет чи видалити. Також є можливість налаштувати класи, які будуть використовуватись в процесі розмітки.

Налаштувавши середовище для розмітки, далі потрібно обрати частину для роботи та відкрити її в режимі annotation. В цей момент відкривається головне меню обробки фото, в якому є все необхідне для створення обмежувальних рамок у об’єктів зазначеного класу (рис. 4.10).

Рис. 4.10. Робоче вікно розмітки

У даному вікні виконана розмітка за допомогою чотирикутника, яка обмежує об’єкт двома точками, тобто береться верхній лівий кут та правий нижній. Після знаходження всіх об’єктів на зображеннях пакету, потрібно повторити все те ж з іншими частинами. На виході ми отримуємо готовий до роботи датасет.

Наступним кроком для роботи є підготовка даних для навчання та валідаціїї отриманих на виході ваг моделі. Для даного процесу був написаний скрипт, який переводить набір даних у потрібний формат для передачі в модель під час тренування. Анотації для набору даних, який використовується, дотримуються формату COCO, який є дуже популярним форматом. Розмітка в цьому форматі зберігається в json-файлі і має структуру дерева, де є прив’язка до зображення. Кожен елемент набору даних називається sample, в собі зберігає шлях до файлу, метадані про фото та всю необхідну розмітку. Приклад відображення sample можна побачити на рис. 4.11.

Рис. 4.11. Відображення Sample з усією інформацією всередині

Розмітка зображення відображення в полі detections->Detection і містить в собі id, назву та обмежувальну рамку, нормалізовану по відношенню до розміру фото.

Для навчання моделі YOLOv5 необхідно перевести розмітку з json-файла у .txt файли для кожного зображення, де кожен рядок описує обмежувальну рамку.

```
0  0.480109 0.631250 0.692969 0.713278
0  0.741016 0.522222 0.314844 0.933333
```

Рис. 4.12. Приклад необхідного для тренування формату розмітки

На рис. 4.12 відображено необхідний формат для тренування. Першою цифрою виступає номер класу, за технічним завданням він один, тому там буде тільки 0. Далі йде чотири цифри, що вказують на розташування обмежувальної рамки відносно зображення, тобто перші дві цифри – це координати центру рамки, а два наступних – це ширина та висота рамки (рис. 4.13). Координати нормалізовані та приведені до формату від 0 до 1.

Рис. 4.13. Приклад опису обмежувальних рамок

Для формування необхідного формату було написано скрипт по обробці вхідних даних, що на виході дає необхідні дані для тренування. Важливо отримати дві папки:

- labels, де зберігається кожен txt-файл з обмежувальними рамками;
- images, де лежать створенні посилання на оригінальні фото з відповідними іменами до txt файлів.

4.5. Конфігурація файлів для навчання

Наступним кроком перед тренування є налаштування всіх конфігураційних файлів. Перший з яких містить шляхи до тренувального та валідаційного датасету, а також інформацію про наявні класи та їх кількість, що можна побачити на рис. 4.14.

```
# train and val data as 1) directory: path/images/,
train: /Users/mmasniy/banners_dataset/train/images
val: /Users/mmasniy/banners_dataset/val/images

# number of classes
nc: 1

# class names
names: ['front_sign']
```

Рис. 4.14. Конфігураційний файл banners_dataset.yaml

Наступний файл для налаштування – це конфіг з гіперпараметрами моделі. Ось список всіх використаних гіперпараметрів у навчанні моделі: lr0: 0.0032, lrf: 0.12, momentum: 0.843, weight_decay: 0.00036, warmup_epochs: 2.0, warmup_momentum: 0.5, warmup_bias_lr: 0.05, box: 0.0296, cls: 0.243, cls_pw: 0.631, obj: 0.301, obj_pw: 0.911, iou_t: 0.2, anchor_t: 2.91, anchors: 3.63, fl_gamma: 0.0, hsv_h: 0.0138, hsv_s: 0.664, hsv_v: 0.464, degrees: 0.373, translate: 0.245, scale: 0.898, shear: 0.602, perspective: 0.0, flipud: 0.00856, fliplr: 0.5, mosaic: 1.0, mixup: 0.243.

Це стандартний набір гіперпараметрів та їх значень для навчання YOLOv5 на COCO датасеті, на них модель була преднавчена.

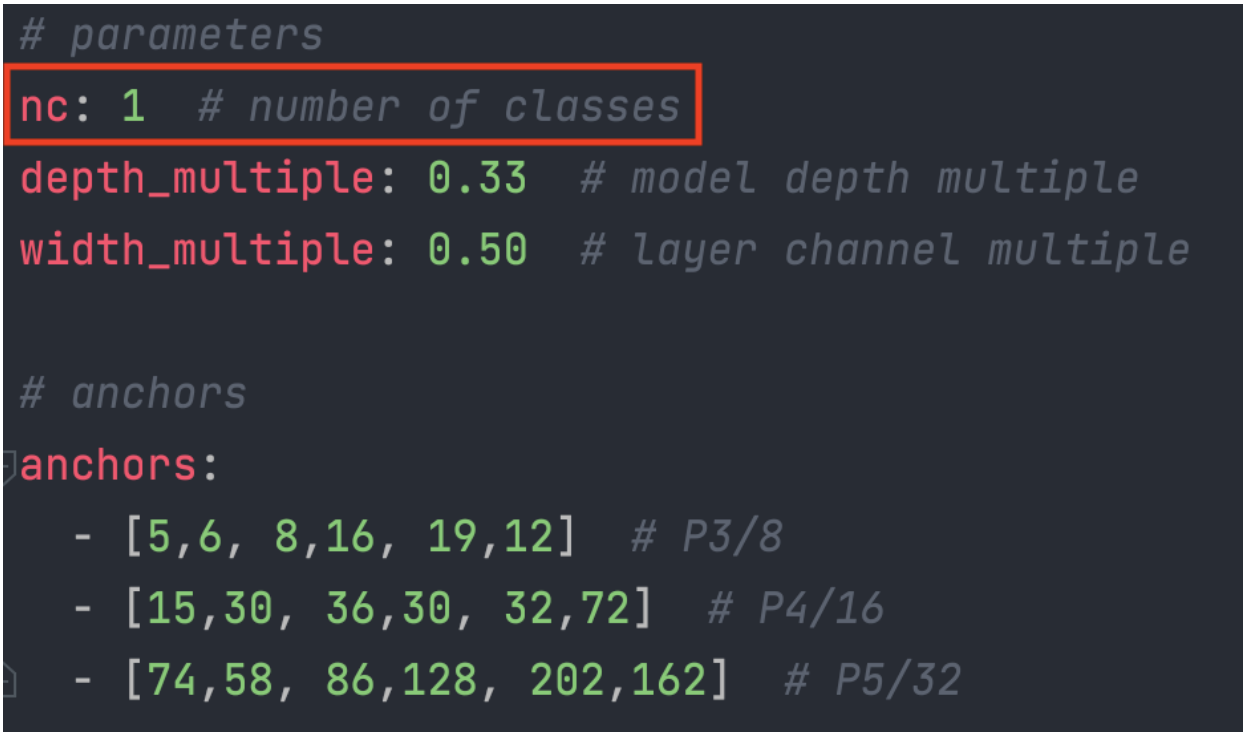


Рис. 4.15. Заміна необхідного елементу в архітектурі CNN

Не менш важливим в процесі підготовки є файл налаштування архітектури згорткової нейронної мережі YOLOv5, саме в ньому вказані всі частини з яких складається модель. До змін підлягає тільки частина, де вказується кількість класів, які буде приймати мережа (рис. 4.15), повна архітектура мережі знаходиться в додатку Д.

Тепер необхідно розібрати всі необхідні параметри, що передаються при навчанні. Розглядатись будуть параметри, що приймає в себе скрипт для тренування – train.py.

При навчанні моделі потрібно передати параметри до скрипта, який використовує всю надану інформацію та проводить тренування. Опцій є дуже багато, проте використовуються найосновніші:

- img - розмір зображення. Зображення квадратне. Розмір оригінального зображення змінюється зі збереженням співвідношення сторін. Довша сторона зображення змінюється до цього числа. Коротша сторона підбита сірим кольором (рис. 4. 16);

Рис. 4.16. Приклад форматування зображення перед тренуванням

- batch – розмір пакета, які будуть передаватись нейронці під час навчання;
- epochs – кількість епох для тренування нейронної мережі;
- data - YAML-файл даних, який містить інформацію про набір даних - шлях до зображень та мітки;
- workers – кількість ядер CPU, які будуть задіяні;
- cfg – шлях до файлу з архітектурою;
- weights – попередньо підготовлені ваги, з яких буде починатись тренування, також є можливість тренування з нуля, необхідно передати в параметр пустий рядок: **--weights ' '**;
- name – назва папки з результатами навчання, наприклад журнали тренувань та вихідні навчені ваги. Інформація буде зберігатись в папці під назвою ./runs/train/name;
- hyp - файл YAML, який описує вибір гіперпараметрів, якщо не вказано, використовується файл data/hyp.scratch.yaml.

Повністю сформована команди запуску тренування нейронної мережі на рис. 4.17.

```
!python train.py --img 640 --batch 16 --epochs 100 --data /content/train-yolov5/data/banners_dataset.yaml \
--cfg /content/train-yolov5/models/custom_yolov5s.yaml --weights ../yolov5s.pt --name yolov5s_results
```

4.17. Формування команди для навчання моделі

4.6. Ітераційне навчання та валідація моделі YOLOv5

Для початку було розглянуто два варіанта навчання згорткової нейронної моделі YOLOv5:

1. Перший – це навчати модель з «нуля», тобто буде створено ваги на основі архітектури з нульовими коефіцієнтами на вузлах.

2. Другий – використати попередньо навчену модель на датасеті COCO. Ці ваги сформовані та мають певні заключення про пошук 80ти класів об’єктів на фото (рис. 4.17).

teacher	student	mAP@.5	mAP@.5:.95	precision	recall
None	yolov5n	0.405	0.237	0.525	0.39
yolov5n	yolov5s	0.435	0.256	0.568	0.412

Рис. 4.18. Результат оцінки моделі за основними параметрами

Тому маючи ваги з певним результатом було б не доцільно навчати модель з повного «нуля». Було застосовано архітектуру YOLOv5s та обрано ваги, що були преднавчені на COCO datasets.

Наступними кроками буде ітераційне тренування мережі з використанням створеного кастомного навчального набору даних. Процес навчання буде складатись з поступового навчання та валідації кожного кроку. Необхідно розбити тренувальний набір даних на частини.

Для отримання поступового результату, було отримано три частини тренувальних даних та одна валідаційна. В абсолютних числах – це train в розмірі тисяча, дві та чотири тисячі зображень та val – одна тисяча фото.

При запуску тренування моделі командою з рис. 4.17. ми отримаємо такий вивід до терміналу. На рис. 4.19 можна побачити ініціалізацію ваг архітектури YOLOV5s, де передається структура та встановлюються відповідні гіперпараметри.

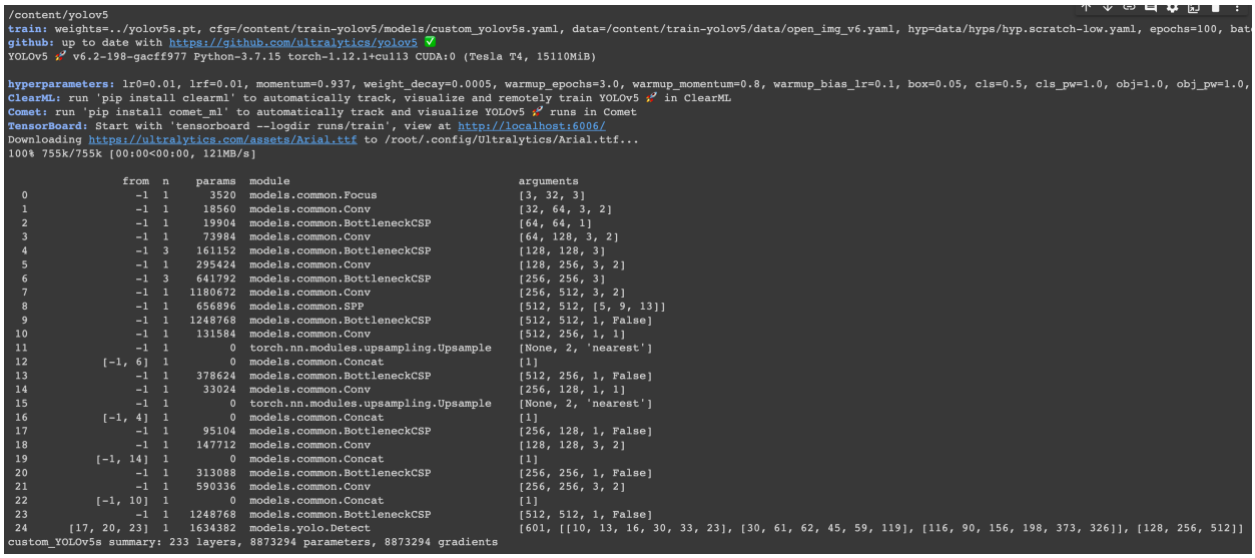


Рис. 4.19. Ініціалізація мережі при тренуванні

Після формування архітектури, встановлюються шляхи до навчальних та валідаційних даних. Також налаштовуються якорі для тренування CNN (рис. 4.19).

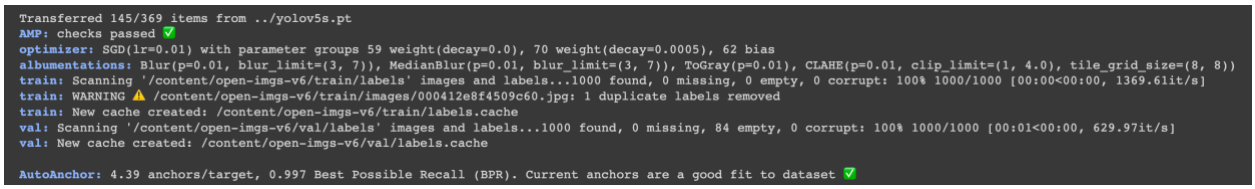


Рис. 4.20. Відображення встановлення шляхів до навчальних даних та налаштування якорів

Налаштувавши середовище, програма запускає тренування моделі на 100 епох з розміром пакетів даних в 16 фото та за розміром 640 х 640 пікселів.

Наступний рис. 4.20 відображає процес навчання YOLOv5s на створеному датасеті. На кожній ітерації можна побачити результати проміжних етапів тренування моделі. Зазвичай вони не дуже інформативні і направлені на надавання інформації про конкретний пакет начальних даних, що не дає підстав робити заключення про вдалий тест чи ні.

```
Plotting labels to runs/train/yolov5s_results/labels.jpg...
Image sizes 640 train, 640 val
Using 2 dataloader workers
Logging results to runs/train/yolov5s_results
Starting training for 100 epochs...

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       0/99      5.53G     0.1041     0.1066      0.142        160        640: 100% 63/63 [01:05<00:00,  1.04s/it]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 32/32 [00:22<00:00,  1.40it/s]
                   all       1000       6909          0          0          0          0

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       1/99      8.25G     0.1001     0.1124      0.141         89        640: 100% 63/63 [01:01<00:00,  1.03it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 32/32 [00:23<00:00,  1.36it/s]
                   all       1000       6909          0          0          0          0

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       2/99      8.25G    0.09805     0.1121     0.1394         96        640: 100% 63/63 [01:02<00:00,  1.00it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 32/32 [00:21<00:00,  1.48it/s]
                   all       1000       6909          0          0          0          0

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       3/99      9.33G    0.09557     0.1112     0.1363        122        640: 100% 63/63 [01:00<00:00,  1.04it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 32/32 [00:23<00:00,  1.39it/s]
                   all       1000       6909   5.73e-05   0.000588   3.03e-05   1.04e-05

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       4/99      9.33G    0.09183     0.1072     0.1288        178        640: 100% 63/63 [00:59<00:00,  1.05it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 32/32 [00:24<00:00,  1.29it/s]
                   all       1000       6909   0.000299    0.00209   0.000179   3.71e-05

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       5/99      9.33G    0.08954     0.1086     0.1195        353        640: 100% 63/63 [00:59<00:00,  1.05it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 32/32 [00:26<00:00,  1.21it/s]
                   all       1000       6909    0.00015    0.00497   0.000111   2.62e-05

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       6/99      9.33G    0.08808     0.1156     0.1126        217        640: 100% 63/63 [01:00<00:00,  1.04it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 32/32 [00:25<00:00,  1.25it/s]
                   all       1000       6909   0.000194    0.00655    0.00022    6.7e-05
```

Рис. 4.21. Відображення тренування та проходження навчальних епох

По завершенню навчання відображається загальна інформація про тренування, а саме вказується кількість пройдених епох, час тренування та вказуються шляхи до вихідних ваг моделі (рис. 4.21). Є два вихідні файли best.pt та last.pt. Алгоритм адаптований для зберігання найкращої моделі в навчані та останньої.

```
      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      99/99      9.33G    0.05925    0.09184    0.06805        107        640: 100% 63/63 [00:58<00:00,  1.07it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 32/32 [00:24<00:00,  1.31it/s]
                   all       1000       6909      0.556     0.0152     0.0103    0.00445

100 epochs completed in 2.417 hours.
Optimizer stripped from runs/train/yolov5s_results/weights/last.pt, 18.1MB
Optimizer stripped from runs/train/yolov5s_results/weights/best.pt, 18.1MB
```

Рис. 4.22. Результат навчання моделі

Після першої ітерації навчання було проведено порівняння результату, який був збережений за шляхом ./runs/train/yolov5s_results/weights/best.pt, де знаходиться найкращий варіант навчених ваг та попередніх, що були відображені на рис. 4.17.

teacher	student	mAP@.5	mAP@.5:.95	precision	recall
None	yolov5n	0.405	0.237	0.525	0.39
yolov5n	yolov5s	0.435	0.256	0.568	0.412
yolov5s	yolov5_custom_1k	0.728	0.515	0.924	0.649

Рис. 4.23. Порівняння best.pt та попередніх варіантів ваг

З результату порівняння можна зробити заключення, що навчання моделі на створених даних дає гарний приріст до робото-спроможності моделі. Проте також варто переглянути метрики на графіках для всіх тренувальних епох, щоб переконатись в правильності побудованого заключення (рис. 4.23).

Рис. 4.24. Графіки метрик навчання моделі

Маючи результати головних метрик за тренуванням, можна підтвердити, що тренування приносить достатньо хороший результат. Тому є підстави стверджувати, що набір даних сформований достатньо добре і дає

можливість отримати потрібний результат на виході. Тому було проведено ще два тренування і валідація результуючих ваг (рис. 4.24).

Рис. 2.25. Результати тренування нейронки на всьому тренувальному наборі даних

Після отримання результуючих ваг згорткової нейронної мережі YOLOv5s, було проведено тестування з «прогону» валідаційних даних через мережу (рис. 4.25).

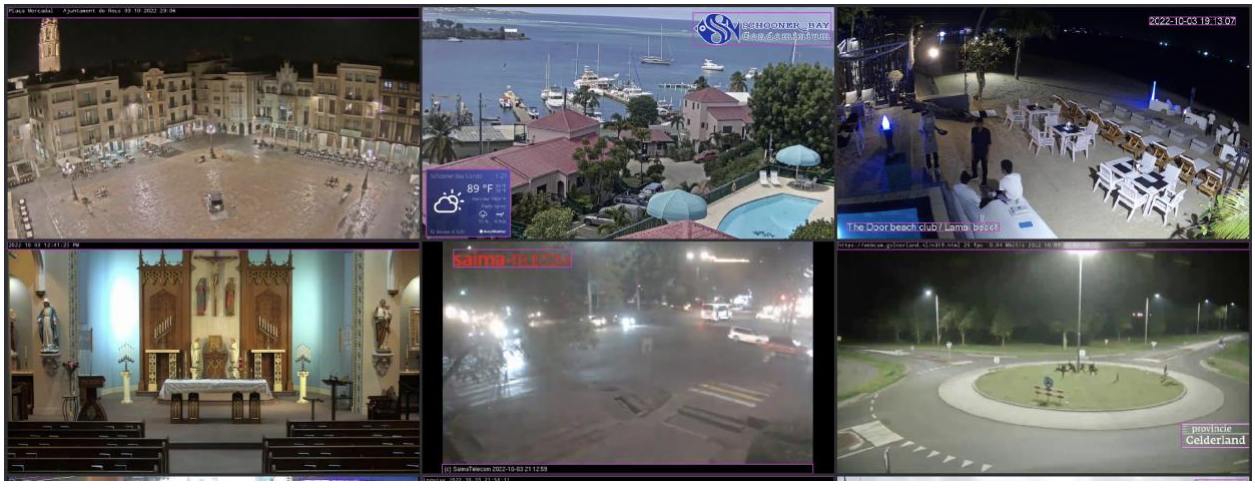


Рис. 4.26. Результат навченої нейронної мережі

З тесту проведеному на валідайному датасеті можна зробити заключення, що поставлену задачу з навчання мережі на створених тренувальних даних – виконано.

ВИСНОВКИ ДО РОЗДІЛУ 4

Навчання нейронної мережі YOLOv5s складається з поступових кроків, при виконанні яких можна отримати бажаний результат у знаходженні об’єктів. Перед тим, як виконувати певні кроки з навчання мережі, варто поставити конкретну ціль, що має бути на виході. Далі, відштовхуючись від задач, які мають виконуватись, необхідно виділити можливі ракурси, місця, вид, час доби і тд., що будуть описувати реальні умови роботи нейронки і тільки після цього намагатись підготувати перші ітераційні датасети.

Кожне створення набору даних несе в собі багато важливих дій, які заключаються в знаходженні джерела даних, наприклад, використано загально доступні сайти youtube.com та google.com, а далі у зборі обраних видів зображення.

З результатів отриманих під час виконання кваліфікаційної роботи можна зробити заключення, що дані мають бути не тільки якісними та позиційними для моделі, а й кількісними, тобто хороший датасет складається з різноманітних зображень великої кількості.

Також важливим фактором ще є розмітка отриманих даних, яка несе прямий вплив на фінальний результат. Створення обмежувальних рамок має бути повністю у відповідності до заданих класів. Це дає можливість нейронці створити правдиві заключення про об’єкт.

Тренування згорткових нейронних мереж має бути дотримуватись певних правил, а особливо у випадку тренування на нових створених наборів даних. Важливо виконувати вище описані пункти для отримання якісної згорткової нейронної мережі для знаходження об’єктів у реальному часі.

ВИСНОВКИ

В епоху розвитку комп'ютерних технологій, все дедалі частіше відбувається розробка нових винаходів, які об'єднують людину та машину, не виключенням стало і розвиток технологій в сфері штучного інтелекту та машинного навчання. В останні декілька десятків років, все більше розробників займаються розвитком технологій пов'язаних зі створення «особистості» у вигляді машини. Поки це не реалізовано повноцінно, проте кожен новий крок дає багато цікавих проектів та новітніх розробок, що допомагає спростити монотонну роботу людей. Тому всі підприємства або великі корпорації намагаються автоматизувати максимально процеси обробки рішень машиною, яка діє без емоцій та втоми, що допомагає уникнути мілких, проте дуже болючих помилок для росту компанії.

Однією з таких сфер для оптимізації є комп'ютерний зір, де система на основі отриманих просторових зображень виділяє признаки та приймає рішення для подальших дій. Цей метод набирає все більшої популярності, тому що може бути застосований майже у будь-якій сфері, прикладом може слугувати система розпізнавання ракових пухлин на основі МРТ-знімків та аналізів пацієнта або ж розпізнавання порушення правил автомобілями на дорозі загального користування.

Проте розробки в даній сфері продовжуються і дедалі більше цікавих проектів для покращення процесів певної сфери діяльності людини. Саме для автоматизація рутинних завдань в роботі інженера даних і є метою даного наукового дослідження.

Під час аналізу предметної області, було про розглянуто принципи роботи та навчання штучного інтелекту, його оптимізацію та можливі проблеми в процесі використання.

Проаналізовано принцип роботи комп'ютерного зору як окремої галузі штучного інтелекту та створених реалізацій для вирішення поставленої цілі.

Проведено дослідження та пошук архітектури згорткової нейронної мережі задля отримання найкращих результатів розробки на виході. В даному випадку, вибір архітектури має важливе значення для подальшої роботи, тому

що вибір оптимального рішення призводить до росту продуктивності системи в цілому. Розглянуто принципи роботи обраної архітектури моделі YOLO та принципів взаємодії нейронів, що об’єднані в шари для покращення продуктивності.

Під час написання кваліфікаційної роботи досліджено сучасні шаблони навчання нейронних мереж, їхні архітектурні рішення, шаблони проектування додатків з використанням CNN та стек сучасних технологій для взаємодіх з моделями штуного інтелекту.

Розкрито та описано проблематику створення наборів навчальних та тестувальних даних, сферу діяльності інженерів даних. Описано задачі, для вирішення яких використовується комп’ютерний зір, та як їх вирішувати. Відображено головне поняття в роботі з фото – ядро та представлено модель, на якій базується робота з зображеннями.

Для розробки системи використано модель згорткових нейронних мереж сімейства YOLO, фреймворк для навчання PyTorch, бібліотека для роботи з датасетами FiftyOne та додаток для розмітки об’єктів на фото CVAT.

Систему треновано та протестовано ітераційно, що показало принцип покращення роботи моделі за рахунок збільшення навчальних даних. Під час роботи були усунуті проблеми в функціонуванні моделі, що відображались під час навчання.

Всі поставленні завдання виконано. Результатом виконання кваліфікаційної роботи є система для знаходження банерів та написів зроблених поверх зображення. Дана модель відповідає описаним та поставленим вимогам, повністю протестована та готова до інтегрування в робочі процеси створення навчальних даних в подальшому.

Отримана CNN має перспективу поширюватися на ринку IT-технологій за рахунок зручності, доступності та важливості. Створена система є адаптивною та гнучкою - це означає, що можливе її майбутнє вдосконалення новими функціями для створення додаткових можливостей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Solem J.E. Programming Computer Vision with Python: Tools and algorithms for analyzing images / J.E. Solem. - New York : O'Reilly Media, 2018. – 243 s.: – Bibliogr.: s. 29-44.

2. Simon J.D. Prince Computer Vision: Models, Learning, and Inference / J.D. Simon Prince. – London: Cambridge University Press, 2017. – 598 s.: – Bibliogr.: s. 19-25.

3. МАШИННЕ НАВЧАННЯ ПРОСТИМИ СЛОВАМИ. [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: http://mmf.lnu.edu.ua/en/aren/1739

4. Szeliski R. Computer Vision: Algorithms and Applications (Texts in Computer Science) Second Edition / R. Szeliski - Berlin: Spriger, 2022. – 947 s. : – Bibliogr.: s. 98-114.

5. Elgendy M. Deep Learning for Vision Systems / M. Elgendy. – New York: Manning Publications, 2020. – 480s.: – Bibliogr.: s. 146-183.

6. Levine D. WHITTLE'S Gait Analysis 5's EDITION / D. Levine, J.Richards, M.W. Whittle. – London: Churchill Livingstone, 2012. – 192 s.: .: – Bibliogr.: s. 145-173.

7. Average graphics card prices have halved since the start of 2022. [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://www.pcgamer.com/average-graphics-card-prices-have-halved-since-the-start-of-2022

8. Moroney L. AI and Machine Learning for Coders: A Programmer's Guide to Artificial Intelligence / L. Moroney. - New York : O'Reilly Media, 2020. – 392 s. .: – Bibliogr.: s. 185-213.

9. A Review of the Evolution of Vision-Based Motion Analysis and the Integration of Advanced Computer Vision Methods Towards Developing a Markerless System. [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://sportsmedicine-open.springeropen.com/articles/10.1186/s40798-018-0139-y

10. NBA G League and Second Spectrum partner for cutting-edge data tracking [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://pr.nba.com/nba-g-league-second-spectrum/

11. Deep Learning For Computer Vision: What It Is And How Companies Can Use It [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://geniusee.com/single-blog/deep-learning-for-computer-vision

12. Deep Learning: GoogLeNet Explained [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://towardsdatascience.com/deep-learning-googlenet-explained-de8861c82765

13. Very Deep Convolutional Networks for Large-Scale Image Recognition [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://arxiv.org/abs/1409.1556v6

14. Deep Residual Learning for Image Recognition [Електроний ресурс]: [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://arxiv.org/abs/1512.03385

15. Xception: Deep Learning with Depth-wise Separable Convolutions [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://iq.opengenus.org/xception-model/

16. YOLO — You only look once, real time object detection explained [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006

17. You Only Look Once: Unified, Real-Time Object Detection [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://arxiv.org/pdf/1506.02640.pdf

18. Big Data To Good Data: Andrew Ng Urges ML Community To Be More Data-Centric And Less Model-Centric [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://analyticsindiamag.com/big-data-to-good-data-andrew-ng-urges-ml-community-to-be-more-data-centric-and-less-model-centric/

19. YOLO V1 Architecture [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://medium.com/@ankushsharma2805/yolo-v1-v2-v3-architecture-1ccac0f6206e

20. Object Detection Explained: YOLO v2. [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://medium.com/mlearning-ai/object-detection-explained-yolo-v2-3e3086789ffb

21. Object Detection Explained: YOLO v3. [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://pr.nba.com/nba-g-league-second-spectrum/

22. YOLO v4 explained in full detail. [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://medium.com/aiguys/yolo-v4-explained-in-full-detail-5200b77aa825

23. COCO Dataset [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://cocodataset.org/#home

24. Augmentation for small object detection [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://www.researchgate.net/publication/338171640_Augmentation_for_small_object_detection

25. YOLOv5 : The Latest Model for Object Detection [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://medium.com/axinc-ai/yolov5-the-latest-model-for-object-detection-b13320ec516b

26. YOLO Explained [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://medium.com/analytics-vidhya/yolo-explained-5b6f4564f31

27. Why can’t we find enough Data Engineers? [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://medium.datadriveninvestor.com/why-cant-we-find-enough-data-engineers-dbc0d053208b

28. What Is the Difference Between a Data Engineer, a Data Scientist, and a Data Analyst? [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://www.propeldata.com/blog/what-is-the-difference-between-a-data-engineer-a-data-scientist-and-a-data-analyst

29. What is computer vision? [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://www.ibm.com/topics/computer-vision

30. Classification, Object Detection and Image Segmentation [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/learning-resources/image-segmentation-deeplab-neural-processing-sdk/classification-object-detection-segmentation

31. Image Kernels [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://medium.com/practical-data-science-and-engineering/image-kernels-88162cb6585d

32. Introduction to Computer Vision & OpenCV in Python ? [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://medium.com/analytics-vidhya/introduction-to-computer-vision-opencv-in-python-fb722e805e8b

33. How to Prepare Data for Object Detection? [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://towardsdatascience.com/how-to-prepare-data-for-object-detection-34750c4d00da

34. Computer Vision: Insights from Datatonic's Experts [Електроний ресурс]: [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://medium.com/datatonic/computer-vision-insights-from-datatonics-experts-48e69fdebf88

35. Data Engineering Challenges and How to Overcome Them [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://www.velvetech.com/blog/data-engineering-challenges/

36. How AI and Computer Vision Shape Our World [Електроний ресурс]: [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://www.spiceworks.com/tech/artificial-intelligence/guest-article/how-ai-and-computer-vision-shape-our-world/

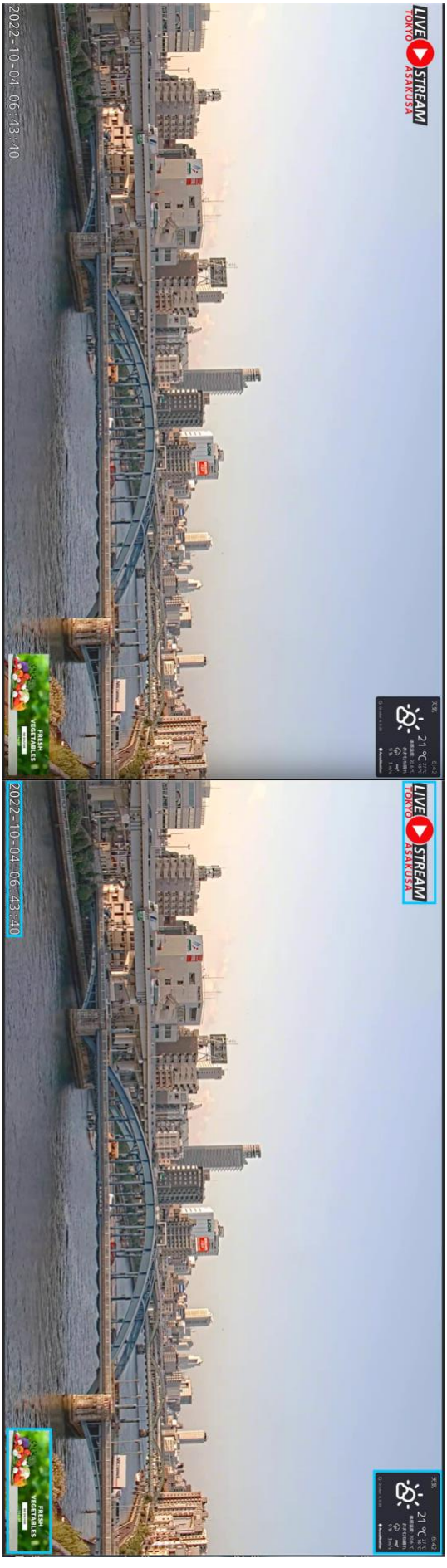
37. Documentation FiftyOne [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://voxel51.com/

38. Computer Vision Annotation Tool [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://www.cvat.ai/

39. OpenCV [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://opencv.org/

40. Tensorflow or PyTorch : The force is strong with which one? [Електроний ресурс] : [Веб-сайт]. - [Електронні дані]. - Режим доступу: https://medium.com/@UdacityINDIA/tensorflow-or-pytorch-the-force-is-strong-with-which-one-68226bb7dab4

ДОДАТОК А. ЦІЛЬ НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ



ДОДАТОК Б. СКРИПТ ДЛЯ ЗБЕРЕЖЕННЯ ФОТО З YOUTUBE.COM

```
import time

from pytube import Channel, YouTube
from pytube.exceptions import LiveStreamError
from concurrent.futures import ThreadPoolExecutor
def get_links_to_live_stream(path_to_txt_file: str) -> list:
    links = list()

    with open(path_to_txt_file, "r") as file:
        for idx, link in enumerate(file.readlines()):
            if link:
                links.append(link.strip())
    return links
def check_live_stream(video) -> list:
    result = []
    try:
        f = video.streams
    except LiveStreamError as e:
        print(video.watch_url)
        result.append(video.watch_url)
    except KeyError as e:
        print("Unknown urls:", video.watch_url)
    except ConnectionResetError as e:
        print(e)

    return result
if __name__ == '__main__':
    urls = []
    live_streams = []
    CHANNELS =
get_links_to_live_stream("/Users/mmasniy/PycharmProjects/cv_dataset_scripts/datasets_mining
/links_to_channels.txt")
    start = time.time()

    for url_channel in CHANNELS:
        channel = Channel(url_channel)
        for idx, video in enumerate(channel.videos):
            urls.append(video)
    print("Len of urls:", len(urls))

    with ThreadPoolExecutor(max_workers=7) as executor:

 for result in executor.map(check_live_stream, urls):
   live_streams.extend(result)
    print(live_streams)
    print("End:", time.time() - start)
    print("All urls:", len(live_streams))
```

ДОДАТОК В СКРИПТ ДЛЯ ЗБЕРЕЖЕНННЯ ФОТО З GOOGLE.COM

```
import io
import os
import time
import shutil
import requests
import argparse
from PIL import Image
from selenium import webdriver
from multiprocessing import Pool
from selenium.webdriver.common.by import By
from bs4 import BeautifulSoup as BeautifulSoup
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
def save_image(data: tuple) -> None:
    url, path_to_save = data
    try
        # first layer for downloading img
        response = requests.get(url, stream=True, timeout=5)
        with open(path_to_save, 'wb') as file:
            shutil.copyfileobj(response.raw, file)
        try:
            test_open = Image.open(path_to_save)
        except Exception as e:
            print(e)
            # second layer for downloading img
            image_file = io.BytesIO(response.content)
            # open img with PIL
            image_file.seek(0)
            image = Image.open(image_file)
            # paste image in size 1920x1080 to keep proportions
            # image = PIL.ImageOps.contain(image, (1920, 1080), Resampling.LANCZOS)
            image.save(path_to_save)
    except Exception as e:
        print(e)
        print(url)
    time.sleep(1)
def find_urls(data: tuple) -> set:
    url, path_to_driver = data
    # print("Start process to search:", url)
    unique_ulrs = set()
    options = Options()
    options.add_argument("--no-sandbox")
    options.add_argument("--headless")
    driver = webdriver.Chrome(service=Service(path_to_driver), options=options)
    driver.get(url)
    driver.maximize_window()
    time.sleep(1)
    for i in range(20):
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(1)
```

```
    try:
        driver.find_element(By.CSS_SELECTOR, '.mJxzWe').click()
    except Exception as e:
        print(e)
    soup = BeautifulSoup(driver.page_source, "html.parser")
    images = soup.findAll("a", "wXeWr islib nfEiy")
    for i in range(0, len(images)):
        try:
            imgs = driver.find_elements(By.CSS_SELECTOR, f'#islrg > div.islrc > div:nth-
child({i}) > a.wXeWr.islib.nfEiy')

            for idx, img in enumerate(imgs):
                try:
                    img.click()
                    soup_step = BeautifulSoup(requests.get(img.get_attribute("href")).content,
"html.parser")
                    down_imgs = soup_step.findAll("img")
                    for down_img in down_imgs:
                        unique_ulrs.add(down_img["src"])
                        time.sleep(1)
                except Exception as e:
                    #
driver.get_screenshot_as_file(f"/home/mmasnyi/Desktop/google_search/media/screen_{i}_{idx}
.png")
                    # print(img)
                    continue
        except Exception as e:
            # print(e)
            continue
    print("Unique  links by url:", len(unique_ulrs))
    driver.close()
    driver.quit()
    return unique_ulrs
def get_unique_data_from_file(filepath: str) -> set:
    unique_data = set()

    with open(filepath, "r") as file:
        for line in file.readlines():
            if line.strip():
                unique_data.add(line.strip())
    return unique_data
if __name__ == "__main__":
    """
    python3 download_images_from_google.py --source
/home/mmasnyi/Desktop/google_search/search_links.txt \
    --unique-links /home/mmasnyi/Desktop/google_search/unique_links_CCTV_1812.txt \
    --chromedriver /home/mmasnyi/Desktop/chromedriver --treads 5 \
    --download-folder /home/mmasnyi/Desktop/google_search
    """
    parser = argparse.ArgumentParser(description='Scrape Google images')
```

```
    parser.add_argument('--chromedriver', type=str, default="", help="Path to chromedriver file")
    parser.add_argument('-tr', '--treads', default=1, type=int, help="Number of treads to save img")
    parser.add_argument('-s', '--source', type=str, required=True, help="Path to file with search
links")
    parser.add_argument('-f', '--download-folder', required=True, type=str,
default="/home/mmasnyi/",
                        help="Path to download files")
    parser.add_argument('-ul', '--unique-links', type=str, default="./default_name.txt",
                        help="File with unique urls img")
    args = parser.parse_args()
    print("Initialisation environment!")
    all_img = set()
    unique_links_this_session = list()
    search_links = set()
    if not os.path.isdir(args.download_folder):
        os.makedirs(args.download_folder)
    all_img = get_unique_data_from_file(args.unique_links)
    search_links = get_unique_data_from_file(args.source)
    print("Prepare data for searching links")
    search_data = zip(search_links, [args.chromedriver] * len(search_links)
print("Run Pool to search with processes:", args.treads)
    with Pool(processes=args.treads) as executor:
        for result in executor.map(find_urls, search_data):
            unique_links_this_session.extend(result)
    unique_links_this_session = set(unique_links_this_session).difference(all_img)
    print("All unique images:", len(all_img))
    print("Unique images of this session:", len(unique_links_this_session))
    save_path = os.path.join(args.download_folder, "data")
    if not os.path.isdir(save_path):
        os.makedirs(save_path)
    data = []
    print("Prepare data for saving imgs by links")
    for idx, item in enumerate(unique_links_this_session):
        name = f"google_search_{idx + len(all_img)}.jpg"
        path = os.path.join(save_path, name)
        data.append((item, path))
    with Pool(processes=args.treads) as executor:
        executor.map(save_image, data)
    all_img = all_img.union(unique_links_this_session)
    with open(os.path.join(args.download_folder, f"unique_links_CCTV_{len(all_img)}.txt"),
"w") as file:
        for link in all_img:
            file.write(f"{link}\n")
```

ДОДАТОК Г. СКРИПТ ДЛЯ ІНТЕГРАЦІЇ З CVAT.AI

```
import argparse
import glob, os
import fiftyone as fo
import time
NAME = 'banners_dataset'
SUBNAME = 'train'
BASE_DIR = "/home/mmasnyi"
DATASET_DIR = os.path.join(BASE_DIR, NAME, SUBNAME)
LABELS_DIR = os.path.join(BASE_DIR, NAME, SUBNAME)
DATASET_EXPORT = 1
key = NAME + SUBNAME
CLASSES = ["person", "car", "pet"]
tmp_len = 70000
SUB_LEN = 300
def import_base_dataset():
    dataset = fo.Dataset.from_dir(
        dataset_type=fo.types.COCODetectionDataset,
        data_path=os.path.join(DATASET_DIR, 'data'),
        labels_path=os.path.join(LABELS_DIR, 'labels_filtered.json'))
    return dataset
def CVAT_show_dataset():
    dataset = import_base_dataset()
    session = fo.launch_app(dataset)
    session.wait()
def CVAT_delete_dataset():
    a = fo.list_datasets()
    print("list of datasets:", a)
    for b in a:
        dataset = fo.load_dataset(b)
        dataset.delete()
    print("deleted:", dataset.deleted)
    print("list of datasets:", fo.list_datasets())
def CVAT_upload():
    dataset = import_base_dataset()
    dataset_len = len(dataset)
    dataset_parts = int(dataset_len / SUB_LEN)
    if dataset_len % SUB_LEN != 0:
        dataset_parts += 1
    print("Dataset len:", dataset_len, "parts:", dataset_parts)
    for i in range(dataset_parts):
        if (i * SUB_LEN) + SUB_LEN > dataset_len:
            sub_max = dataset_len
        else:
            sub_max = (i * SUB_LEN) + SUB_LEN
        dataset_new = dataset[i * SUB_LEN:sub_max].select_fields("ground_truth").clone()
        dataset_new.name = NAME + SUBNAME + '_part_' + str(i)
        dataset_new.persistent = True
        dataset_new.annotate(anno_key + '_part_' + str(i), label_field="ground_truth",
url="http://13.36.168.203:8080/",
                             project_name=NAME)
```

```
        print("Uploading dataset:", NAME + SUBNAME + '_part_' + str(i))
        time.sleep(2)
    print("list of datasets:", fo.list_datasets())
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description="Work with CVAT")
    parser.add_argument('-i', '--input', required=True, help="Path to folder with img")
    parser.add_argument('-l', '--labels', required=True, help="Path to file with labels")
    parser.add_argument('-cfg', '--config', type=str, required=True, help="Path to config file")
    parser.add_argument('-yt', '--youtube', type=str, default="", help="File's path with links from
youtube")
    parser.add_argument('-m', '--model', type=str, required=True, help="Path to folder with model
for auto-label")
    parser.add_argument('-o', '--output-path', type=str, required=True, help="Path to folder with
output images")
    parser.add_argument('-ph', '--path-to-heatmaps', type=str, required=True, help="Path to folder
with heatmap files")
    parser.add_argument('-d', '--set-device', type=int, default=0, choices=[0, 1], help="Set GPU's
index by 0 or 1")
    parser.add_argument('--cpu', action="store_true")
    parser.add_argument('--heatmap', action="store_true",
                        help="filter to detect intersection the center of the object on heatmap")
parser.add_argument('--intersection-objects', action="store_true",
                        help="filter to detect intersections objects on frame")
    args, unknown = parser.parse_known_args()
    CVAT_show_dataset()
    CVAT_delete_dataset()
    CVAT_upload()
```

ДОДАТОК Д. АРХІТЕКТУРА YOLOV5

```
# parameters
nc: 1  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple

# anchors
anchors:
  - [5,6, 8,16, 19,12]  # P3/8
  - [15,30, 36,30, 32,72]  # P4/16
  - [74,58, 86,128, 202,162]  # P5/32

# YOLOv5 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Focus, [64, 3]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, C3, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 9, C3, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, C3, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 3, C3, [1024, False]],  # 8
  ]

# YOLOv5 head
head:
  [[-1, 1, Conv, [512, 1, 1]], # 9
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]],  # cat backbone P4
   [-1, 3, C3, [512, False]],  # 12

   [-1, 1, Conv, [256, 1, 1]], # 13
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 4], 1, Concat, [1]],  # cat backbone P3
   [-1, 3, C3, [256, False]],  # 16 (P3/8-small)

   [-1, 1, Conv, [256, 3, 2]],
   [[-1, 13], 1, Concat, [1]],  # cat head P4
   [-1, 3, C3, [512, False]],  # 19 (P4/16-medium)

   [-1, 1, Conv, [512, 3, 2]],
   [[-1, 9], 1, Concat, [1]],  # cat head P5
   [-1, 3, C3, [1024, False]],  # 22 (P5/32-large)

   [[16, 19, 22], 1, Detect, [nc, anchors]],  # Detect(P3, P4, P5)
  ]
```